PERANCANGAN DAN IMPLEMENTASI APLIKASI PETA *CONTOUR* DENGAN MENGGUNAKAN ALGORITMA KRIGING PADA SUATU LAPANGAN MINYAK X

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

JON MARYONO
10145019230

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2009

PERANCANGAN DAN IMPLEMENTASI APLIKASI PETA KONTUR DENGAN MENGGUNAKAN ALGORITMA KRIGING PADA SUATU LAPANGAN MINYAK X

JON MARYONO
10145019230

Tanggal Sidang : 20 Februari 2009
Periode Wisuda : Juli 2009

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Pengelolaan sumur minyak perlu dilakukan perencanaan yang matang dan dilakukan secara profesional dengan cara menganalisa sumur minyak dengan memetakan kontur cadangan minyak pada suatu lapangan minyak yang menggambarkan besar produksi sumur-sumur minyak.

Dalam pembuatan peta kontur diperlukan data-data sumur minyak untuk membentuk titik-titik kontur minyak. Masalah yang timbul dalam pembentukan peta kontur adalah bagaimana memprediksi titik-titik kontur dari sumur-sumur minyak yang datanya tidak diketahui sehingga dapat dibentuk peta kontur secara keseluruhan dari suatu lapangan minyak. Oleh karena itu, diperlukan algoritma kriging untuk memprediksi semua titik-titik kontur pada suatu lapangan minyak.

Aplikasi ini dikembangkan dengan menggunakan bahasa pemrograman Java dan database MySQL akan dapat membantu dalam memprediksi sumur-sumur minyak yang memiliki tingkat produksi yang besar, sumur-sumur baru yang belum dikelola, dan besar cadangan minyak pada sumur minyak karena didalam peta kontur dapat dilihat besar kandungan minyak berdasarkan tingkatan warna yang mewakili ketinggian kandungan minyak.

Hasil akhir dari sistem ini adalah grafik peta kontur lapangan minyak yang dapat membantu para analis perminyakan dalam menganalisa lokasi sumur-sumur penghasil minyak yang lebih baik.

Kata kunci : Analisa sumur minyak, Java, Kriging, Peta kontur

DESIGN AND IMPLEMENTATION APPLICATION OF CONTOUR MAP USING KRIGING ALGORITHM IN AN OIL FIELD X

JON MARYONO
10145019230

Date of Final Exam : *February 20th 2009*
Graduation Ceremony Priod : *July 2009*

Information of Technology Engineering Departement
Faculty of Sciences and Technology
State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Management of oil wells needed good planning and professionally done by analyzing the contour map of oil wells with oil reserves in an oil field that describes the large production of oil wells.

Contour map needed oil wells data to form oil contour points. Problems arising out in making contour map is how to prediction the contour points from the oil wells which data unknown so it can making all contour map from an oil field. Therefore, kriging algorithms needed to predict all contour points in an oil field.

This application is developed using Java programming language and MySQL database will be helpful in predicting the oil wells that have a greater level of production, new wells that have not been managed, and large reserves of oil in oil wells because in the contour map seen major oil content based on the level of color that represent the height of the oil content.

As the result of this system is a graph of oil field contour maps that can help analysts to analyze the location of petroleum in oil-producing wells better.

Keywords : Contour map, Java, Kriging, Oil analyst.

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Provinsi Riau memiliki potensi sumber daya mineral yang cukup besar, salah satunya adalah sumber daya minyak. Potensi daerah Riau sebagai wilayah ladang minyak menjadikan Riau salah satu Provinsi terkaya di Indonesia. Hal ini menunjukan betapa pentingnya minyak bagi perekonomian di Riau. Minyak merupakan sumber daya mineral yang sangat penting karena paling banyak dipakai bagi kehidupan manusia dalam melakukan aktifitasnya sehari-hari. Krisis melambungnya harga minyak dapat mempengaruhi ekonomi suatu wilayah, negara, atau bahkan dunia Oleh karena itu dalam pengelolan minyak haruslah memiliki perencanaan yang matang dan dilakukan secara profesional agar kinerja pengelolaan minyak dilakukan secara lebih optimal.

Minyak mentah atau *petroleum* merupakan percampuran antara pentana dan hidrokarbon yang lebih berat, yang diambil dari lapisan tanah melalui sumur minyak yang didalamnya mengandung lapisan reservoir minyak mentah. Reservoir minyak bisa menghasilkan hidrokarbon cair yang lebih ringan seperti propana dan butana, yang digolongkan dalam gas alam cair (LNG). Komposisi minyak mentah bervariasi dari satu ladang minyak ke ladang minyak lainnya yang dapat dilihat berdasarkan ketebalan lapisan reservoirnya.

Kesalahan dalam eksplorasi sumur minyak, dimana sumur minyak yang dieksplorasi diketahui memiliki produksi minyak mentah yang sedikit padahal

terdapat sumur minyak lain yang masih berada di ladang yang sama memiliki produksi yang lebih besar dapat menyebabkan kerugian baik dari segi biaya, sumber daya, dan waktu bagi suatu perusahaan minyak. Hal ini disebabkan oleh beberapa faktor, salah satunya adalah kesalahaan awal analisa dalam pengelolaan sumur minyak berdasarkan hasil produksi minyaknya. Untuk mengatasi hal ini diperlukan suatu analisa yang tepat dan akurat dengan teknologi informasi agar dapat mengetahui sumur mana yang paling menghasilkan dalam suatu lapangan minyak agar produksi minyak yang dikelola perusahaan menjadi lebih besar dan biaya yang dikeluarkan menjadi lebih sedikit.

Analisa terhadap kandungan cadangan minyak pada suatu lapangan dapat dilakukan dengan memetakan kontur cadangan minyak pada suatu lapangan minyak yang mengambarkan besar produksi sumur-sumur minyak. Peta *Contour* adalah peta yang memiliki informasi tentang ketinggian permukaan tanah pada suatu tempat terhadap permukaan laut yang digambarkan dengan garis-garis kontur. Informasi kontur yang terdapat pada peta kontur dapat digunakan untuk membuat model grafik, peta dua dimensi (2D), dan tiga dimensi (3D), sehingga untuk menganalisa suatu peta kontur lapisan *reservoir* minyak dapat lebih mudah dilakukan Dari peta kontur tersebut dapat dilihat ketebalan lapisan reservoir dan hubungan antar sumur minyak yang menjadi dasar sumur minyak mana yang produksi dan kandungan minyak mentahnya yang besar maupun kecil berdasarkan warna-warna tertentu yang mewakili ketebalan *reservoir*.

Untuk membentuk peta kontur diperlukan data produksi yang dipergunakan untuk membentuk titik-titik kontur cadangan minyak pada suatu

lapangan. Namun untuk menganalisa suatu lapangan minyak diperlukan metode untuk memprediksi titik kontur sumur baru pada suatu lapangan minyak yang belum dikelola dan tidak diketahui besar kandungan minyak sehingga dapat dibentuk peta kontur cadangan pada suatu lapangan minyak.

Adapun cara untuk menentukan sumur-sumur baru dengan hasil yang optimum , diantaranya dengan menggunakan metoda-metoda geostatistik yang digunakan untuk menaksir lokasi tak bersampel adalah kriging. Metoda ini cukup baik karena bersifat BLUE (*Best Linier Unbiased Estimation*) atau perkiraan sampel terbaik. Data yang digunakan meliputi data hasil uji produksi yaitu misalnya produksi minyak, produksi air, tekanan, atau temperatur yang ditampilkan sebagai peta, misalnya peta kontur. Dengan menggunakan algoritma kriging titik-titik kontur cadangan minyak dapat diprediksi dan pembuatan peta kontur suatu lapangan minyak dapat dengan mudah dilakukan.

Dari kondisi diatas penulis melakukan penelitian tentang bagaimana memetakan kontur ketebalan *reservoir* atau lapisan *reservoir* dan hubungan antar sumur minyak dengan menggunakan algoritma kriging pada suatu lapangan minyak.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat dirumuskan suatu masalah yaitu bagaimana membuat aplikasi untuk "**Peta *Contour* dengan menggunakan algoritma kriging pada suatu lapangan minyak**". Sebagai sampel atau contoh kasus adalah pemetaan kontur kandungan minyak pada suatu lapangan minyak X.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Pemetaan *Contour* hanya berbasis 2D (2 Dimensi).
2. Tipe Algoritma kriging yang dipakai adalah *Punctual (Ordinary)* Kriging.
3. Indikator yang digunakan dalam perhitungan algoritma kriging hanya berupa data produksi minyak.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut :

1. Membangun suatu aplikasi peta *Contour* data sumur minyak dengan menggunakan algoritma kriging untuk memberikan suatu informasi prediksi kandungan minyak pada suatu lapangan minyak sebagai bahan analisa dalam pengelolaan suatu lapangan minyak.
2. Mempelajari dan menerapkan algoritma kriging untuk kebutuhan pemetaan kontur, visualisasi, manajemen data, dan analisa sumur-sumur penghasil minyak yang lebih baik pada suatu lapangan minyak.

1.5 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi enam bab yang masing-masing bab telah dirancang dengan suatu tujuan tertentu, berikut penjelasan masing-masing bab :

BAB I PENDAHULUAN

Membahas tentang deskripsi umum dari tugas akhir ini yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan serta sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini menjelaskan tentang konsep berorientasi objek, gambaran umum perminyakan, peta kontur, algoritma kriging, dan tinjauan umum tentang Java.

BAB III METODE PENELITIAN

Membahas mengenai langkah-langkah yang dilakukan dalam penelitian dan metode analisis yang akan digunakan,.

BAB IV ANALISA DAN PERANCANGAN

Membahas mengenai analisa sistem, analisa perhitungan kriging, UML (*Unified Modelling Language*) serta perancangan sistem.

BAB V IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dibahas mengenai implementasi perangkat lunak, lingkungan implementasi, pengujian perangkat lunak, hasil pengujian dan kesimpulan pengujian.

BAB VI PENUTUP

Menjelaskan tentang kesimpulan dan saran dari pembuatan tugas akhir ini.

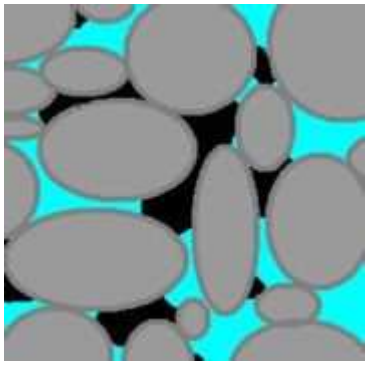
BAB II

LANDASAN TEORI

2.1 Gambaran Umum Perminyakan

Geologi minyak bumi adalah salah satu cabang ilmu geologi untuk mengetahui keberadaan minyak bumi di bawah tanah, kemudian mengeksplorasi dan memproduksinya (*id.wikipedia.org, 2008*). Secara umum ada dua jenis geologi minyak bumi, yaitu geologi eksplorasi minyak bumi yang mencakup pencarian minyak bumi dan geologi produksi minyak bumi. Produksi minyak bumi dalam bidang perminyakan bukan diartikan untuk membuat minyak bumi, tetapi hanyalah membuat fasilitas untuk mengalirkan minyak bumi dari bawah tanah ke atas permukaan tanah, dengan menggunakan pemboran dan pompa-pompa melalui sumur-sumur minyak.

Teori keberadaan minyak bumi ada dua buah, yaitu teori organik dan teori anorganik. Teori organik sekarang ini banyak dianut oleh para ahli geologi, dimana minyak bumi dipercayai dihasilkan oleh sisa-sisa organisme yang sudah mati berjuta-juta tahun yang lalu. Sedangkan teori anorganik kebanyakan berkembang di Eropa Timur dan Rusia di mana para ahli mempercayai bahwa minyak bumi dapat dihasilkan bukan dari bahan organik. Prinsip geologi minyak bumi yang sekarang umum dipakai adalah teori organik sehingga minyak bumi sering disebut bahan bakar fosil. Bila teori anorganik terbukti, maka akan muncul lagi sumber-sumber minyak bumi yang selama ini belum dieksplorasi.



Gambar 2. 1 Pori batuan

Dari gambar diatas dapat dilihat warna abu-abu adalah pasir, biru adalah air, dan hitam adalah minyak.

2.1.1 Kajian Geologi

Secara ilmu geologi, untuk menentukan suatu daerah mempunyai potensi akan minyak bumi, maka ada beberapa kondisi yang harus ada di daerah tersebut , yaitu sebagai berikut (*id.wikipedia.org, 2008*) :

1. Batuan Sumber (*Source Rock*)

   Yaitu batuan yang menjadi bahan baku pembentukan hidrokarbon. biasanya yang berperan sebagai batuan sumber ini adalah serpih. batuan ini kaya akan kandungan unsur atom karbon **(C)** yang didapat dari cangkang - cangkang fosil yang terendapkan di batuan itu.

2. Tekanan dan Temperatur

   Untuk mengubah fosil tersebut menjadi hidrokarbon, tekanan dan temperatur yang tinggi di perlukan. Tekanan dan temperatur ini akan mengubah ikatan kimia karbon yang ada dibatuan menjadi rantai hidrokarbon.

3. Migrasi

   Hirdokarbon yang telah terbentuk dari proses di atas harus dapat berpindah ke tempat dimana hidrokarbon memiliki nilai ekonomis untuk diproduksi. Di batuan sumbernya sendiri dapat dikatakan tidak memungkinkan untuk di ekploitasi karena hidrokarbon di sana tidak terakumulasi dan tidak dapat mengalir. Sehingga tahapan ini sangat penting untuk menentukan kemungkinan eksploitasi hidrokarbon tersebut.

4. *Reservoar*

   Adalah batuan yang merupakan wadah bagi hidrokarbon untuk berkumpul dari proses migrasinya. Reservoar ini biasanya adalah batupasir dan batuan karbonat, karena kedua jenis batu ini memiliki pori yang cukup besar untuk tersimpannya hidrokarbon. Reservoar sangat penting karena pada batuan inilah minyak bumi di produksi.

5. Perangkap (*Trap*)

   Sangat penting suatu reservoar di lindungi oleh batuan perangkap. tujuannya agar hidrokarbon yang ada di reservoar itu terakumulasi di tempat itu saja. Jika perangkap ini tidak ada maka hidrokarbon dapat mengalir ketempat lain yang berarti ke ekonomisannya akan berkurang atau tidak ekonomis sama sekali.

Jika semua kriteria di atas terpenuhi maka daerah tersebut kemungkinan mempunyai potensi minyak bumi atau pun gas bumi. Sedangkan untuk menentukan ekonomis atau tidaknya diperlukan kajian yang lebih lanjut pada langkah kajian Geofisika.

2.1.2 Kajian Geofisika

Pada tahapan ini metoda - metoda khusus digunakan untuk mendapatkan data yang lebih akurat guna memastikan keberadaan hidrokarbon dan kemungkinannya untuk dapat di ekploitasi. Data-data yang dihasilkan dari pengukuran pengukuran merupakan cerminan kondisi dan sifat-sifat batuan di dalam bumi. Ini penting sekali untuk mengetahui apakan batuan tersebut memiliki sifat - sifat sebagai batuan sumber, reservoar, dan batuan perangkap atau hanya batuan yang tidak penting dalam artian hidrokarbon. Metoda-metoda ini menggunakan prinsip-prinsip fisika yang digunakan sebagai aplikasi engineering, yaitu sebagai berikut (*id.wikipedia.org, 2008*) :

1. Eksplorasi seismic

   Ini adalah ekplorasi yang dilakukan sebelum pengeboran. kajiannya meliputi daerah yang luas. dari hasil kajian ini akan didapat gambaran lapisan batuan didalam bumi.

2. Data resistiviti

   Prinsip dasarnya adalah bahwa setiap batuan berpori akan di isi oleh fluida. Fluida ini bisa berupa air, minyak atau gas. Membedakan kandungan fluida didalam batuan salah satunya dengan menggunakan sifat resistan yang ada pada fluida. Fluida air memiliki nilai resistan yang rendah dibandingkan d engan minyak, demikian pula nilai resistan minyak lebih rendah dari pada gas. dari data log kita hanya bisa membedakan resistan rendah dan resistan tinggi, bukan jenis fluida karena nilai resitan fluida berbeda beda dari tiap daerah.

3. Data porositas.
4. Data berat jenis.

   Data ini diambil dengan menggunakan alat logging dengan bantuan bahan radioaktif yang memancarkan sinar gamma. Pantulan dari sinar ini akan menggambarkan berat jenis batuan.

2.2 Peta *Contour*

Peta *Contour* (Kontur*)* atau dikenal juga dengan Peta Topografi adalah peta yang menggambarkan bentuk relief (tinggi rendahnya) permukaan bumi. Dalam peta topografi digunakan garis kontur (*Contour line*) yaitu garis yang menghubungkan tempat-tempat yang mempunyai ketinggian sama. Pada gambar 2.2 terlihat gambar garis kontur yang membentuk garis yang berbelok-belok dan tertutup serta merupakan rangkaian dari titik-titik. (*romenah, 2008*). Kegunaan peta topografi:

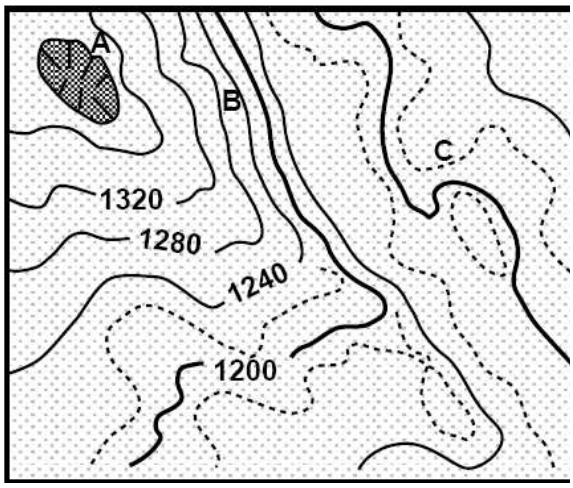
1. mengetahui ketinggian suatu tempat.
2. memperkirakan tingkat kecuraman atau kemiringan lereng.

Ciri utama peta topografi adalah menggunakan garis kontur. Beberapa ketentuan pada peta topografi:

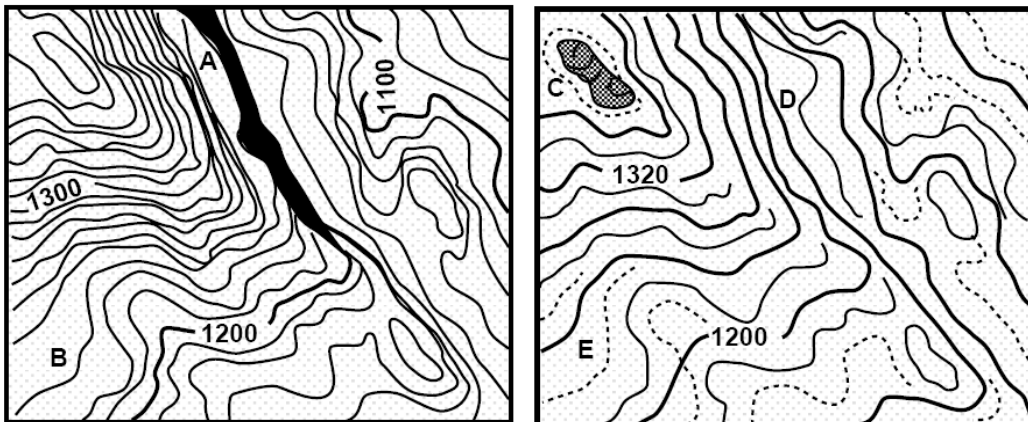
1. Makin rapat jarak kontur yang satu dengan yang lainnya menunjukkan daerah tersebut semakin curam. Sebaliknya semakin jarang jarak antara kontur menunjukkan daerah tersebut semakin landai.
2. Garis kontur yang diberi tanda bergerigi menunjukkan depresi (lubang/cekungan) di puncak, misalnya puncak gunung yang berkawah.

3. Peta topografi menggunakan skala besar, antara 1 : 50.000 sampai 1 : 100.000.

Peta Kontur memiliki atribut yang dibutuhkan berbentuk polygon yang terdiri dari kumpulan titik (*vertex*) yang memiliki koordinat X dan Y. Polygon tersebut dapat digambarkan dengan menambahkan warna-warna tertentu. Untuk menggambarkan bentuk dari peta kontur dapat kita lihat dalam beberapa gambar berikut :



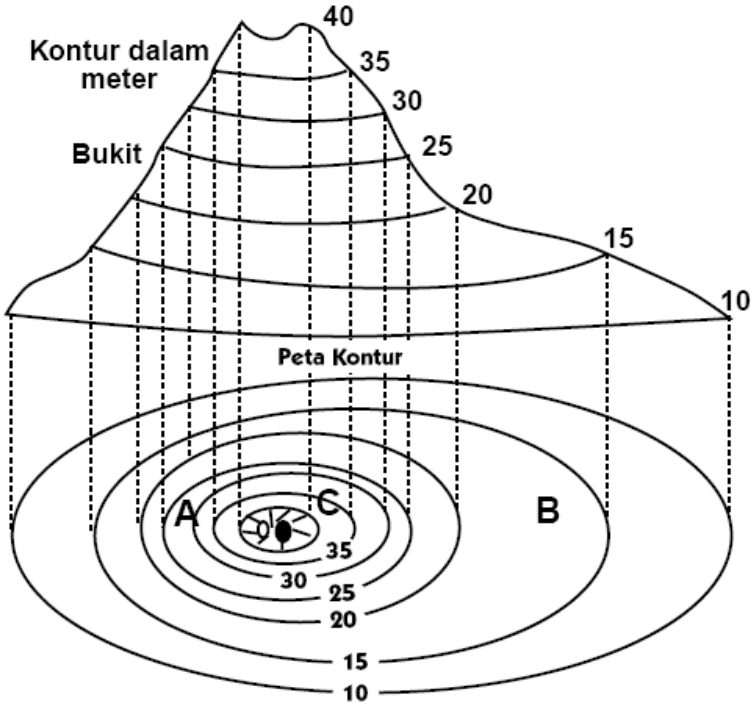
Gambar 2. 2 Garis kontur dengan interval (jarak antara 2 kontur) 40 m (*romenah, 2008*)



Gambar 2. 3 Jarak kontur

Perhatikan gambar 2.3. Berdasarkan jarak antara kontur dan tanda pada kontur, Anda dapat menyimpulkan bahwa: Pada peta kiri, A adalah daerah curam karena jarak antara garis konturnya rapat dan B adalah daerah landai karena jarak konturnya jarang. Sedangkan pada peta 2, D adalah daerah curam karena jarak konturnya rapat, E adalah daerah landai karena jarak konturnya jarang, dan C adalah daerah depresi (lubang/cekungan) di puncak karena diberi tanda bergerigi.

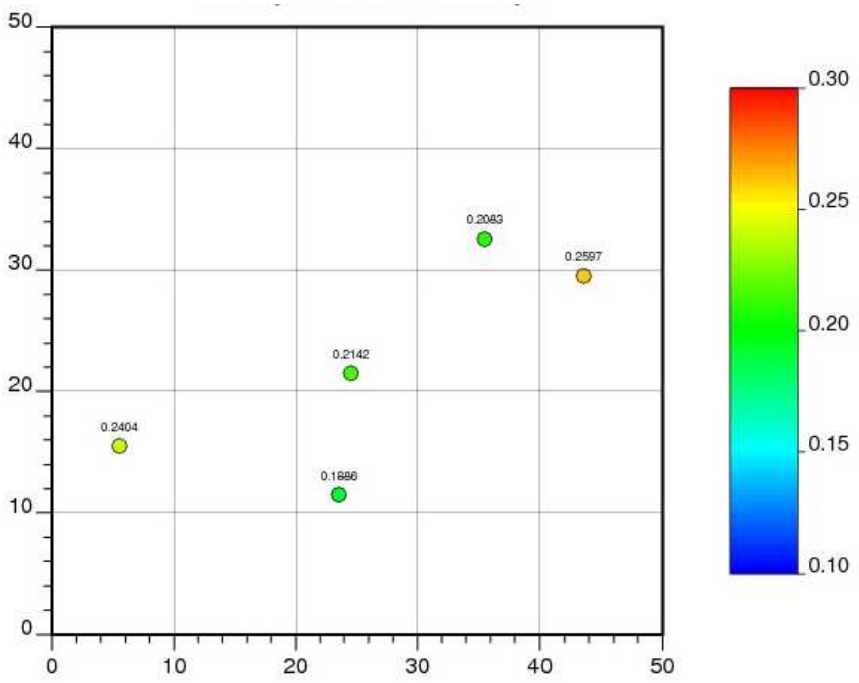
Pada gambar 2.4, menunjukkan kenampakan gunung dengan puncaknya yang digambarkan menjadi peta kontur. Pada gambar tersebut, A daerah curam, B daerah landai dan C daerah cekungan di puncak.



Gambar 2. 4 Perubahan penurunan dari kenampakan akan menjadi peta kontur.

2.3 Algoritma Kriging

Dalam buku Jenkins Glover dan Doney yang berjudul ”*Modeling Methods for Marine Science*” disebutkan bahwa Kriging adalah metode statistic geologi yang digunakan untuk mengestimasi nilai dari sebuah titik atau blok sebagai kombinasi linier dari nilai kontur yang terdapat disekitar titik yang akan diestimasi atau lebih dikenal dengan menemukan *Best Linier Unbiased Estimator* (BLUE). Bobot kriging diperoleh dari hasil variansi estimasi minimum dengan memperluas penggunaan semi-variogram. Estimator kriging dapat diartikan sebagai variabel tidak bias dan penjumlahan dari keseluruhan bobot adalah satu. Bobot inilah yang dipakai untuk mengestimasi nilai dari ketebalan, ketinggian, kadar atau variabel lain (*Glover, 2005*).



Gambar 2. 5 Model penggunaan metode kriging (*Villa, 2007*).

Teori Interpolasi dan Ekstrapolasi Kriging dikembangkan oleh seorang ahli matematika Perancis Georges Matheron berdasarkan disertasi Daniel Gerhardus Krige. Kriging memberikan lebih banyak bobot pada kontur dengan jarak terdekat dibandingkan dengan kontur dengan jarak lebih jauh, Bentuk geometri dari data dan karakter variabel yang diestimasi serta besar dari blok juga diperkirakan merupakan pertimbangan yang penting dalam kriging.

Sifat-sifat Kriging :

1. Struktur dan korelasi variabel melalui fungsi $\gamma(h)$
2. Hubungan geometri relatif antar data yang mencakup hal penaksiran dan penaksiran volume melalui (Si,Sj) (hubungan antar data) dan sebagai (Si,V) (hubungan antara data dan volume)
3. Jika variogram isotrop dan pola data teratur, maka sistem kriging akan memberikan data yang simetris
4. Dalam banyak hal hanya kontur di dalam blok dan di sekitar blok memberikan estimasi dan mempunyai suatu faktor bobot masing-masing nol
5. Dalam hal ini jangkauan radius kontur yang pertama atau kedua pertama tidak mempengaruhi (tersaring).
6. Efek screen ini akan terjadi, jika tidak ada nugget effect atau kecil sekali

   $\varepsilon = C0/C.$
7. Efek nugget ini menurunkan efek screen
8. Untuk efek nugget yang besar, semua contoh mempunyai bobot yang sama.
9. Kontur-kontur yang terletak jauh dari blok dapat diikutsertakan dalam estimasi ini melalui nilai rata-ratanya (*en.wikipedia.org, 2006*).

Kelebihan kriging dari metode geostatistik lain (*Bohling, 2005*):

1. Mengurangi kesalahan dalam perhitungan data yang berhubungan.
2. Memberikan perkiraan kesalahan prediksi.
3. Menilai kesalahan perhitungan (*kriging variance*) berdasarkan simulasi keadaan sebenarnya.

2.3.1 *Contour* dan *Grid*

Nilai grid ($\hat{Z}$) diperkirakan dari nilai titik terdekat dari titik data. Jarak dalam grid (D_{ik}) dapat dihitung dengan persamaan :

$$D_{ik} = \sqrt{(x_{1k} - x_{1i})^2 + (x_{2k} + x_{2i})^2} \quad (2.1)$$

keterangan : x_{1k} dan x_{2k} = nilai X pada k (titik yang dicari)

x_{1i} dan x_{2i} = pada i (data sebenarnya).

Kita dapat menghitung nilai dari titik k yang tidak diketahui ($\hat{z}_k$) dengan membagi nilai yang diketahui pada titik i (Z_i) dengan jarak rata-rata dalam grid kontur ($\min(D_{ik})$) .

$$\hat{z}_k = Z_i \mid \min(D_{ik}) \quad (2.2)$$

Pembobotan digunakan untuk menggambarkan titk-titik grid berdasarkan data sebenarnya. Dengan menggabungkan persamaan 2.1 dan 2.2 dan

menambahkan faktor bobot dari titik grid ($w_i = {}^{1}/_{s_i^2}$), maka akan menghasilkan persamaan berikut :

$$\hat{Z}_k = \frac{\sum_i^N \frac{w_i Z_i}{D_{ik}}}{\sum_i^N \frac{1}{D_{ik}}} \quad (2.3)$$

keterangan : $\hat{z}_k$ = Nilai dari titik k yang tidak diketahui.

w_i = nilai bobot.

Z_i = nilai dari titik yang diketahui.

D_{ik} = per jarak grid.

Dimana k (titik yang dicari) dan i (titik yang diketahui).

2.3.2 Variogram

Variabel *Regionalized* adalah variable yang didistribusikan dalam ruang baik berupa waktu, paremeter, property, dan lain-lain. Tiap-tiap titik dalam ruang $\vec{X}$ memiliki sebuah nilai ($\vec{X} = [x_1, x_2, \ldots, x_d]$, dimana d merupakan dimensi ruang. Variabel *Regionalized* direpresentasikan sebagai dan perkiraan titik grid sebagai $\hat{Z}(\vec{X})$. Karakteristik variabel *Regionalized :*

1. local, random, aspek lain yang berupa variabel acak.

2. Struktur umum merepresentasikan beberapa fungsi.

Semivariance merupakan teori variabel *Regionalized* yang menggunakan hubungan data property misalnya ketebalan untuk mengetahui tingkat hubungan antara titik-titik pada suatu permukaan yang digambarkan dalam suatu grafik yang disebut semivariogram. Dengan menggunakan semivarince kita dapat mengkur derajat hubungan spasial (keruangan) dari berapa data sample. Semivariance anatar titik tergantung jarak antar data sample. Semakin dekat jarak antar titik akan semakin kecil semivariance, dan sebaliknya semaking besar jarak antar titik, maka semakin besar semivariance.

Berdasarkan jarak titik-titik yang teratur (h), semivariance ($\gamma(h)$) dapat memperkirakan jarak yang ada dengan rumus (*Glover, 2005*) :

$$\gamma(h) = \frac{1}{2N_h} \sum_{i=1}^{N_h} (z_i - z_{i+h})^2 \qquad (2.4)$$

Keterangan : Z_i = pengukuran dari varabel regionalized yang diambil dari lokasi i.

Z_{i+h} = pengukuran lain yang diambil dari jarak h interval diluar d

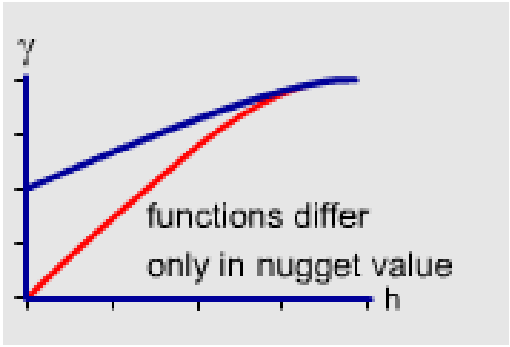
N_h = banyaknya jarak dari titik.

Model ini dapat dikembangkan lagi untuk korelasi (hubungan) antar titik data seperti berikut :

$$\gamma(h) = c_0 + c\left(1 - e^{-\left(\frac{h}{a}\right)}\right) \qquad (2.5)$$

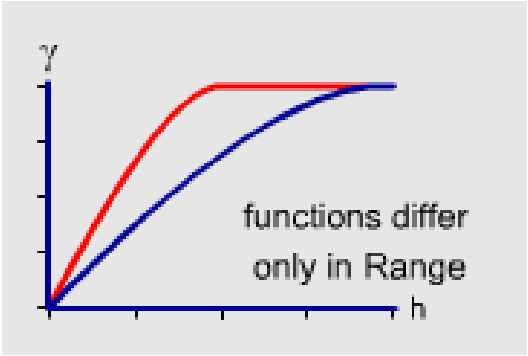
Keterangan :

1. C_0 (*effect nugget*) : merepresentasikan tidak dipecahkan, variasi skala sub grid atau pengukuran error dari variogram.



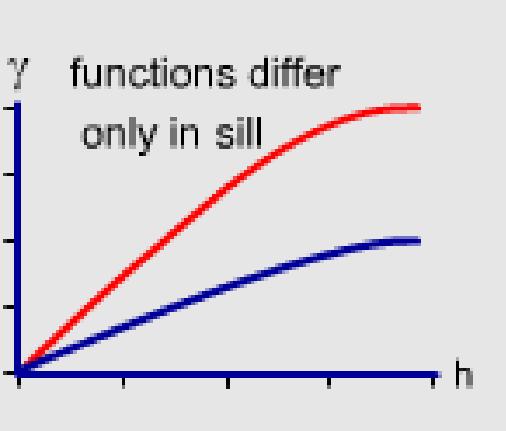
Gambar 2. 6 Bentuk Semivariogram *effect nugget* (*giac.montana.edu, 2008*)

2. α (*range*) : tingkat skala control dari hubungan antara titik data.



Gambar 2. 7 Bentuk Semivariogram *range* (*giac.montana.edu, 2008*).

3. c (*sill*) : jarak daerah semivariance berdasarkan jarak variable *Regionalized.*



Gambar 2. 8 Bentuk Semivariogram *Sill* (*giac.montana.edu, 2008*)

Penggunaan variogram dalam menentukan tingkatan permukaan berdasarkan model variogram yang benar menjadi suatu yang penting. Menggunakan variogram sebagai petunjuk kecenderungan permukaan yang cocok dari data yang bertambah secara berurutan, dapat menentukan beberapa spesifikasi level permukaan yang tidak menyediakan pertambahan statistik yang signifikan dalam data yang ada sampai urutan data terakhir dikurangani 1 (n-1) (*Glover, 2005*).

Analisis permukaan bobot dan lingkungan bergantung sekali dengan *semivariance* data, atau dengan kata lain bergantung fungsi struktur dari data yang ditampilkan. Derajat kesinambungan ruang (spatial continuity) dari data (variable sekitar daerah) memberikan semivariogram dan beberapa bentuk model :

1 . Spherical (Model yang dipakai dalam pembentukan peta kontur minyak).

Perhitungan nilai *sill* lebih cepat dari exponential model, oleh karena itu model variogram ini yang penulis pakai dalam perhitungan kriging.

Secara umum bentuk persamaan sebagai berikut :

$$\gamma(h) = c_0 + c\left(\frac{3h}{2a} - \frac{1h^3}{2a^3}\right) \text{ untuk } h \leq a \quad (2.6)$$

2 . Gaussian

Menampilkan bentuk parabola yang mendekati perkiraan sebenarnya, dengan persamaan sebagai berikut :

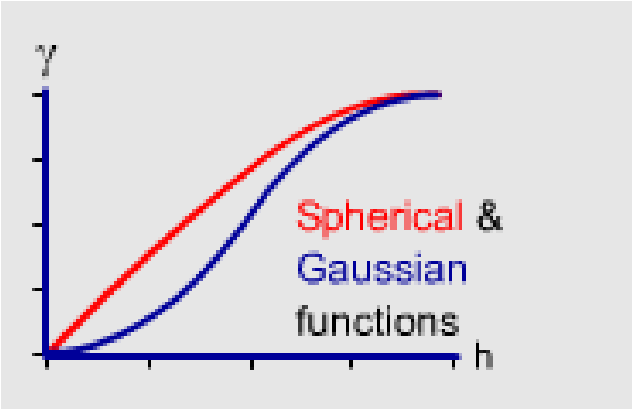
$$\gamma(h) = c_0 + c\left(1 - e^{-\left(\frac{h^2}{a^2}\right)}\right) \quad (2.7)$$

3. *Linear*

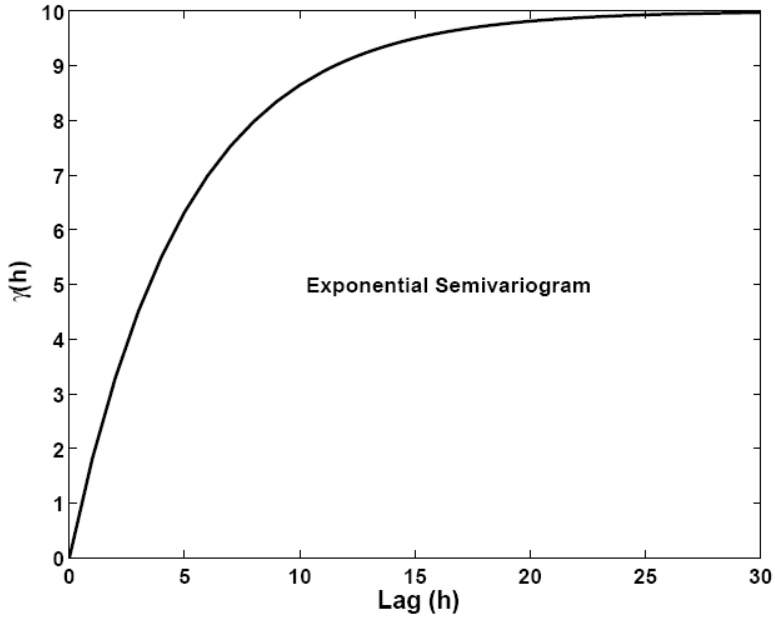
Model dimana data tidak mendukung untuk *sill* dan *range*, dengan persamaan berikut :

$$\gamma(h) = c_0 + bh \quad (2.8)$$

Kecendrungan slope (b) merupakan rasio perbandingan *sill* (C) terhadap *range* (α).



Gambar 2. 9 Bentuk Semivariogram *Sill* (*giac.montana.edu, 2008*)



Gambar 2. 10 Contoh semivariogram dengan *range* 5 dan *sill* 10 (*Glover, 2005*)

2.3.3 Konsep Kriging

Tipe kriging sangat bergantung properti dari nilai acak. Perbedaan dari tipe-tipe kriging terletak bada batasan linear dalam bobot w_i yang menyatakan kondisi unbias; misalnya batasan linear dan metode untuk menghitung bobot. Kriging dapat dibagi menjadi beberapa kelas :

1. *Simple kriging*

   Mengasumsikan batasan yang tidak diketahui menjadi $\mu(x) = 0$.

2. *Ordinary kriging*

   Mengasumsikan batasan yang tidak diketahui menjadi $\mu(x) = \mu$.

3. *Universal kriging* assumes a general linear trend model

   Mengasumsikan batasan yang tidak diketahui menjadi linear umum.

$$\mu(x) = \sum_{k=0}^{P} \beta_k f(x) \tag{2.9}$$

4. *IRFk-kriging*

   Mengasumsikan μ(*x*) menjadi polynomial dalam *x*.

5. *Indicator kriging*

   Mengggunakan fungsi indikator akan memproses dirinya sendiri, yang bertujuan untuk memperkirakan kemungkinan transisi.

6. *Multiple-indicator kriging*

   Indikator kriging bekerja dengan banyak indikator working. *Multiple-indicator kriging* jarang dipakai karena sullit menghubungkan dalam persamaan dan model validasinya.

2.3.4 Ordinary Kriging

Tipe kriging yang dipakai dalam penelitian ini adalah *Ordinary Kriging. Ordinary Kriging* adalah format sederhana dari kriging.

Karakteristik *Ordinary Kriging* :

1. *Ordinary kriging* merupakan tipe kriging yang paling sederhana.
2. Titik dalam dimensi data yang diketahui digunakan untuk memperkirakan titik lain , contohnya adalah titik kontur.
3. Dalam *ordinary kriging*, variable sekitar sangat diperlukan.

$$\hat{Z}_p = \sum \omega_i Z_i \qquad (2.10)$$

Pencarian bobot liner (ω_i) yang terbaik dengan menghitung perkiraan error (ε_p) Z di titik P :

$$\varepsilon_p = (\hat{Z}_p - Z_p) \tag{2.11}$$

Untuk mengetahui perbedaan antara $\hat{Z}_p$ yang telah dihitung dengan keadaan sebenarnya, dimana jumlah sebenarnya (Z_p) tidak diketahui, kita membutuhkan penjumlahan bobot (ω_i) menjadi satu, yang hasilnya berupa perkiraan awal (*unbiased*).

Kita lalu menghitung berbagai error (s_ε) dengan :

$$S_{\hat{\varepsilon}}^2 = \frac{\Sigma(\hat{Z}_p - Z_p)^2}{N} \tag{2.12}$$

Ini hanya memperlihatkan logika yang mendekati sebuah data titik yaitu titik pegangan (*grid*) yang diharapkan untuk memperkirakan bobot yang lebih banyak. Bobot (ω_i) yang digunakan dan perkiraan kesalahan (s_ε) berhubungan dengan $\hat{Z}_p$ melalui semivariogram.

Maka, jika kita mempunyai tiga data titik untuk menghitung satu titik grid, berdasarkan rumus 2.1 kita harus mempunyai persamaan :

$$\hat{Z}_p = \omega_1 Z_1 + \omega_2 Z_2 + \omega_3 Z_3 \tag{2.13}$$

$$\sum_i \omega_i = 1 \tag{2.14}$$

Dengan menggunakan semivariogram diatas kita dapat perhitungan awal :

$$\omega_1\gamma(h_{11}) + \omega_2\gamma(h_{12}) + \omega_3\gamma(h_{13}) = \gamma(h_{1P})$$

$$\omega_1\gamma(h_{21}) + \omega_2\gamma(h_{22}) + \omega_3\gamma(h_{23}) = \gamma(h_{2P}) \qquad (2.15)$$

$$\omega_1\gamma(h_{31}) + \omega_2\gamma(h_{32}) + \omega_3\gamma(h_{33}) = \gamma(h_{3P})$$

Keterangan : $\gamma(h_{ij})$ = semivariance jarak h antara titik control ı dan j.

$\gamma(h_{iP})$ = semivariance dari jarak h antara titik kontrol ı dan titik grid *P*.

Maka akan menghasilkan matriks berikut agar jumlah bobot selalu benar dengan persamaan berikut :

$$\begin{pmatrix} \gamma(h_{11}) & \gamma(h_{12}) & \gamma(h_{13}) & 1 \\ \gamma(h_{21}) & \gamma(h_{22}) & \gamma(h_{23}) & 1 \\ \gamma(h_{31}) & \gamma(h_{32}) & \gamma(h_{33}) & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \lambda \end{pmatrix} = \begin{pmatrix} \gamma(h_{1P}) \\ \gamma(h_{2P}) \\ \gamma(h_{3P}) \\ 1 \end{pmatrix} \qquad (2.16)$$

$$s_\varepsilon^2 = \omega_1\gamma(h_{1P}) + \omega_2\gamma(h_{2P}) + \omega_3\gamma(h_{3P}) + \lambda \qquad (2.17)$$

Keterangan : s_ε^2 = perkiraan error.

ω_1 = bobot yang dicari.

$\gamma(h_{1P})$ = nilai semivariogram

λ = nilai yang menghasilkan penjumlahan bobot menjadi 1

Misalkan kita mendapatkan sampel data seperti berikut :

Tabel 2. 1 Contoh data

Sumur	X	Y	Kandungan minyak(Z)
1	2	4	120
2	6.3	3.4	103
3	2	1.3	142
P	3	3	(tidak diketahui)

Dengan menghitung jarak berdasarkan persamaan 2.1, kita dapat menganalisa struktur analisis semivariogram dengan ketentuan analisa diluar 20 km bernilai nol dan bergeser 4 m2/km, maka menjadi 4*variance jarak (D) yang menghasilkan :

$$\begin{pmatrix} 0 & 13.42 & 11.52 & 1 \\ 13.42 & 0 & 19.14 & 1 \\ 11.52 & 19.14 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \lambda \end{pmatrix} = \begin{pmatrix} 4.00 \\ 13.40 \\ 7.89 \\ 1 \end{pmatrix}$$

Untuk mendapatkan vector maka kita harus membalik matriks (*inverse)* 2.16 untuk mendapatkan nilai ω_1 dan λ yang menghasilkan vector berikut:

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \lambda \end{pmatrix} = \begin{pmatrix} 0.6040 \\ 0.0867 \\ 0.3093 \\ -0.7266 \end{pmatrix}$$

Dari hasil diatas kita dapat menghitung nilai perkiraan menggunakan persamaan 2.13 :

$$\hat{Z}_p = 0.6040(120) + 0.0867(103) + 0.3093(142)$$

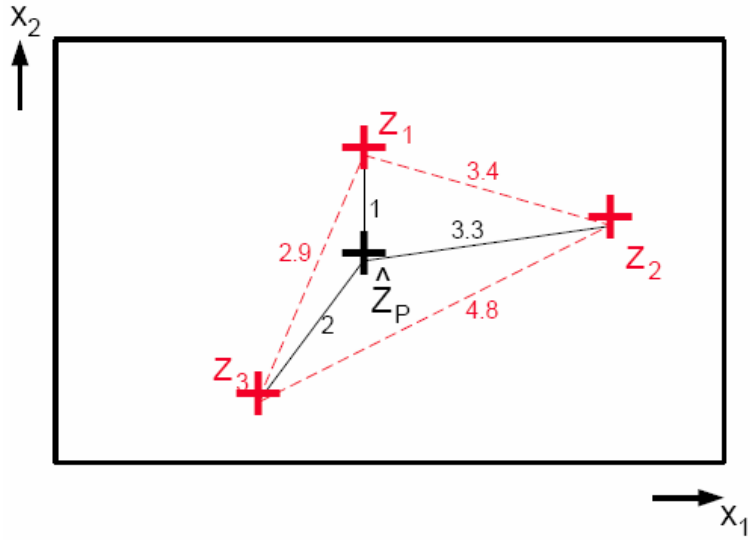
$$= 125.3 \text{ m}$$

Dengan mendapatkan persamaan 2.17 kita dapat menghitung :

$$s_e^2 = 0.6040_1\gamma(h_{1p}) + 0.0867\gamma(h_{2p}) + 0.03093\gamma(h_{3p}) - 0.07266$$

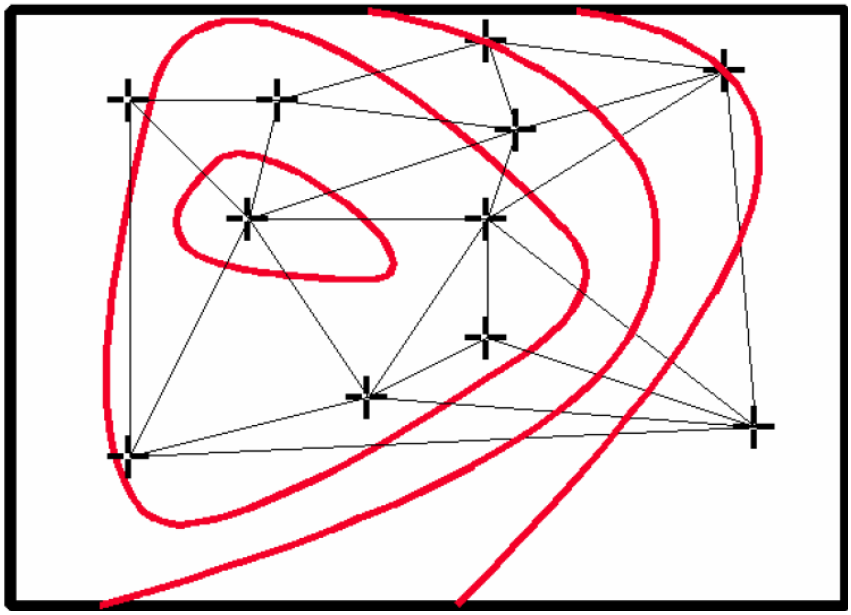
$$= 5.28 \text{ m}^2$$

Hasil diatas menunjukan perkiraan nilai X dititik P adalah 125.3.

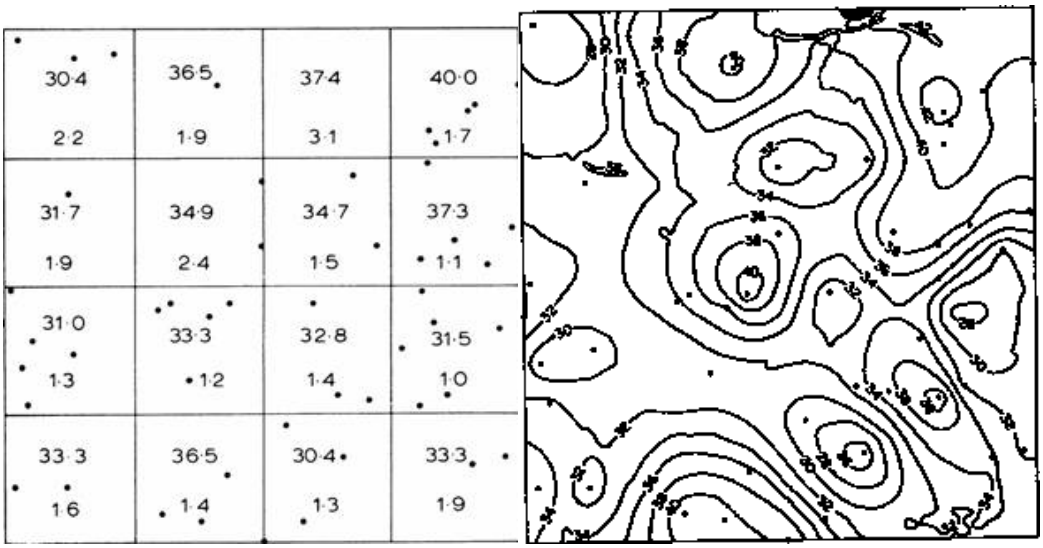


Gambar 2. 11 Layout tiga titik control dan titik grid yang telah dihitung.

Dengan mengambil banyak data sampel yang random, misalnya 50, kita dapat menghasilkan blok-blok yang akan dipergunakan untuk membentuk peta countor.

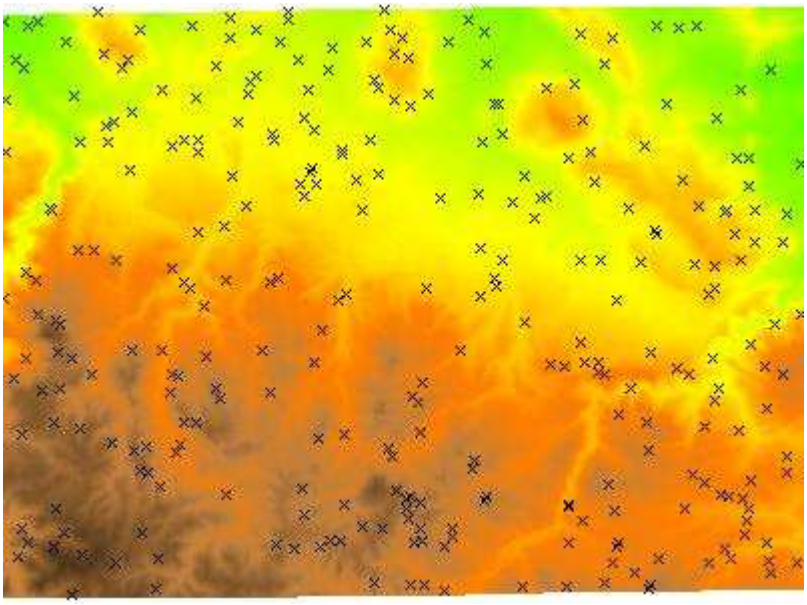


Gambar 2. 12 Menghubungkan titik-titik grid pada peta kontur (*Glover, 2005*)



Gambar 2. 13 Hasil Perhitungan kriging dengan mengambil banyak sampel (kiri) dan hasil peta kontur (kanan) (*Clark, 2008*).

Dengan menambah warna-warna tertentu yang mewakili suatu ketinggian data maka peta Kontur dapat berbentuk sebagai berikut :



Gambar 2. 14 Contoh Hasil Peta Kontur yang menunjukan warna yang diwakili ketinggian data dimana tinggi diwakili warna orange dan rendah warna hijau (*Dylan, 2007*).

2.4 Pendekatan Beorientasi Objek

Metode analisis dibutuhkan sebagai bahan acuan dalam membuat perangkat lunak. Metode yang digunakan adalah pendekatan berorientasi objek (*object oriented*). *Object oriented* (OO) secara umum adalah suatu metodologi/cara yang diambil dari filsafat dunia nyata yang diterapkan pada teknologi informasi, merupakan suatu pola pikir yang diterapkan menyeluruh tentang bagaimana kita memandang sesuatu baik sudut pandang pengguna, pengembang ataupun pengelola teknologi (*Irwan, 2006*).

2.4.1 Prinsip dasar dari *object oriented*

Ada empat prinsip dasar yang utama dari *object oriented*, sebagai berikut :

1. Abstraksi

   Abstraksi adalah pemodelan yang menyangkut aspek yang sangat utama, penting, khusus atau esensial dari sesuatu dengan mengabaikan rincian detail yang kurang atau tidak penting darinya. lainnya.

2. Enkapsulasi

   Enkapsulasi adalah menyembunyikan cara pengimplementasian suatu benda dari pengguna, sehingga pengguna hanya tergantung dan berhubungan dengan antarmuka luarnya saja.

3. Modulariti

   Modulariti adalah memecah sesuatu yang kompleks atau rumit menjadi bagian-bagian kecil yang dapat dikendalikan atau diatur.

4. Hirarki

   Hirarki adalah urutan atau aturan dari tingkatan abstraksi menjadi seperti struktur pohon.

2.4.2 Konsep dasar dari *object oriented*

Ada sembilan konsep dasar dari *object oriented*, sebagai berikut:

1. Objek

   Objek adalah sebuah konsep, abstraksi atau sesuatu yang diberi batasan jelas dan dimaksudkan untuk sebuah aplikasi.

2. Kelas

Kelas adalah deskripsi dari kelompok objek dengan properti yang sama (atribut), kelakuan yang sama (operasi), serta *relationship* dan semantik yang sama.

3. Atribut

   Atribut adalah nama-nama properti dari sebuah kelas yang menjelaskan batasan nilainya dari properti yang dimiliki oleh sebuah kelas tersebut.

4. Operasi

   Operasi adalah implementasi dari layanan yang dapat diminta dari sebuah objek dari sebuah kelas yang menentukan tingkah lakunya.

5. Antar muka (*Polimorpisma*)

   Antar muka adalah sebuah antarmuka yang menutupi bagian-bagian detail didalamnya, disebut juga penerapan dari polimorpisma, yaitu kemampuan untuk menyembunyikan banyak detail implementasi yang berbeda-beda dari dan dengan hanya menggunakan sebuah antar muka yang sama, merupakan juga pengembangan konsep enkapsulasi.

6. Komponen

   Komponen sudah memenuhi persyaratan fungsi yang jelas pada konteks arsitektur yang sudah terbentuk dengan baik.

7. Paket

   Paket adalah mekanisme yang bertujuan umum untuk mengorganisasikan elemen-elemen kedalam sebuah grup.

8. Subsistem

Sebuah subsistem adalah pemodelan elemen yang mempunyai tata bahasa dari paket, seperti dapat terdiri dari pemodelan elemen yang lain dan sebuah kelas, seperti mempunyai tingkah laku sendiri (tingkahlaku dari subsistem dihasilkan dari kelas-kelas atau subsistem lainnya yang dimilikinya).

9. Keterhubungan

   Keterhubungan menyediakan cara-cara berkomunikasi antar objek. Ada beberapa cara keterhubungan antar objek tersebut, yaitu: asosiasi, asosiasi agregasi, asosiasi komposisi, dependensi, generalisasi dan realisasi.

   a. Asosiasi menampilkan keterhubungan struktural antar objek dari kelas yang berbeda, informasi yang harus dipersiapkan untuk jangka waktu tertentu dan keterhubungan dependesi prosedural yang mudah (misalnya, satu objek punya asosiasi yang permanen terhadap objek lainnya).
   b. Dependensi adalah menggunakan keterhubungan yang menampilkan keterhubungan antara pengguna dengan penyedia dimana perubahan spesifikasi pada sisi penyedia akan mempengaruhi pengguna.
   c. Generalisasi adalah keterhubungan membuat khusus ataupun umum dimana elemen-elemen dari elemen yang lebih khusus (subtipe atau *child*) dapat mengganti elemen dari elemen yang lebih umum, misalnya (*parent*).
   d. Realisasi adalah keterhubungan secara tata bahasa antara dua klasifikasi. Satu klasifikasi berlaku sebagai penghubung, sedangkan lainnya menyetujui untuk membawa.

e. Agregasi adalah bentuk asosiasi khusus yang secara kuat memodelkan seluruh bagian dari asosiasi antara hubungan satu bagian kelas secara keseluruhan dengan bagian tertentu dari kelas lainnya.

f. Komposisi adalah bentuk keterhubungan agregasi yang lebih kuat lagi kepemilikannya dan mempunyai jangka waktu yang timbul sesuai kebutuhan.

2.4.3 Analisa *object oriented*

Dalam memulai menentukan langkah untuk menentukan ide dari pembangunan sebuah sistem ataupun menentukan kebutuhan dari sistem ataupun mengelompokkan permasalahan menjadi domain permasalahan yang jelas. Semua cara tersebut adalah langkah pertama yang digolongkan pendekatan permasalahan, apapun nama lainnya langkah pertama inilah yang utama dalam menerapkan *object oriented* dengan semua prinsip dan konsepnya, selanjutnya kita perlu menganalisa hasil pendekatan permasalahan yang ada.

2.4.3.1 Menganalisa permasalahan berdasarkan *object oriented*

Dengan menggunakan analisa *object oriented* kita dapat memodelkan dunia dengan mengidentifikasikan kelas-kelas dan objek-objek, dengan bentuk tersebut akan membentuk keterangan dari semua domain permasalahan.

Pada saat analisa domain permasalahan analisa dengan *object oriented* akan difokuskan pada satu permasalahan spesifik pada satu waktu, kemudian dilanjutkan dengan mencari identifikasi kelas, operasi, struktur dan objek yang umum terhadap semua aplikasi yang akan dikembangkan dari domain

permasalahan yang ada. Seorang sistem analis harus dapat menyatakan apa yang harus dihasilkan dari permasalahan spesifik pada domain, harus ditentukan batasan permasalahannya. Kunci penggunaan ulang perangkat lunak yang dimunculkan pada saat analisa *object oriented* dilakukan pada saat menganalisa domain permasalahan, dan maksudnya adalah penggunaan ulang hasil analisa dan rancangan sebelumnya, bukan menggunakan ulang sebuah kode.

2.4.3.2 Penentuan kebutuhan sistem berdasarkan *object oriented*

Kebutuhan dari sistem dapat ditentukan dilakukan pendekatan terhadap permasalahan dengan menggunakan konsep dasar *object oriented* untuk memformulasikan kebutuhan dari sistem yang dianalisa dan dibangun rancangannya. Ada beberapa aktifitas yang harus dilaksanakan dalam melakukan analisa terhadap domain permasalahan menjadi kebutuhan sistem, pada tahap ini kita juga sudah harus menentukan keterhubungan antar semua kebutuhan dan alur perancangannya.

2.4.3.3 Pemetaan permasalahan dengan *object oriented*

Beberapa konsep secara berurutan yang diterapkan dalam melakukan analisa hasil kebutuhan sistem, guna memetakan permasalahan menjadi spesifik untuk dibuat perancangan desainnya dengan bantuan *pattern*. Bentuk dan fitur dari *pattern*, dan metode serta proses yang digunakan disekitarnya adalah hal yang biasa dalam merancang sebuah arsitektur. *Pattern* dapat dilihat sebagai fitur gambaran penjelasan menyeluruh dari kelas. Dalam memetakan permasalahan, dapat menggunakan semua kemungkinan kelompok konstruksi bahasa yang

formal, semiformal maupun informal. Sebagai contoh: sekelompok kelas *object oriented* dapat direpresentasikan secara tatabahasa, menggunakan peraturan penulisan ulang berdasarkan bentuk tingkatan komposisi *pattern*. Bagaimanapun juga, kita tidak perlu menginterpretasikan kumpulan dari *pattern* atau kelas sebagai sebuah bahasa tersendiri.

2.4.4 Desain *object oriented*

Dalam membahas perancangan menggunakan *object oriented*, langkah pertama yang harus dilakukan adalah bagaimana mendesain hasil pemetaan domain permasalahan yang ada menggunakan *object oriented*. Pada awal bab ini dijelaskan bahwa dalam menerapkan metodologi *object oriented*, harus ditentukan dahulu suatu kesamaan persepsi dalam bentuk bahasa yang mendefinisikan notasi yang sama, saat ini telah dikembangkan suatu bahasa pemodelan untuk metodologi *object oriented* yang menggabungkan hampir semua notasi yang ada menjadi notasi yang standard *Unified Modeling Language* (UML).

2.4.4.1 Mendesain permasalahan menjadi *pattern* desain *object oriented*

Dalam perancangan *object oriented* menerapkan abstraksi dan mekanisme yang menyediakan tingkah laku yang diinginkan oleh model. Perancangan jelas membutuhkan spesifikasi tingkah laku yang tersedia dari lingkungan eksternal dan menambahkan perincian yang dibutuhkan untuk implementasi sistem komputer yang aktual, termasuk interaksi dengan manusia dan aktor lainnya, *task management*, dan managemen data yang terinci.

Aspek internal, adalah lingkungan pandangan pemecahan masalah merupakan penjelasan statis dan dinamis, mempunyai batasan yang jelas, dan aturan yang jelas antar komponen, delegasi, kolaborasi, dan petunjuk, dimana hanya dapat dinyatakan dan dilihat secara notasi (sebagai contoh bila sebuah kelas dinyatakan lagi ada sub kelasnya).

Dalam merancang sistem *object oriented* ada dua konsep dasar yang harus difokuskan, yaitu kelas dan objek. Sebuah kelas yang terbaik juga berbagi properti dengan abstraksi, enkapsulasi, keterbukaan dan keseimbangan yang sesuai. Seperti juga *pattern*, kelas secara normal dapat dibangkitkan, mendukung konstruksi instansiasi berparameter.

2.4.4.2 *Unified Modelling Language* (UML)

Pada tahap perancangan dengan menggunakan metode *object oriented* ini kita harus dapat menterjemahkan hasil analisa dan desain kita yang telah menerapkan bahasa pemodelan yang standard, yaitu UML. UML (*Unified Modelling Language*) adalah suatu bahasa untuk menentukan visualisasi, konstruksi dan mendokumentasikan objek dari sistem *software*, untuk memodelkan bisnis dan sistem *non-software* lainnya (*Suhendar dkk, 2002*).

Tujuan UML :

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.

Faktor pendorong dibuatnya UML :

1. Membangun model untuk suatu sistem *software* sangat bergantung pada konstruksinya atau kemudahan dalam memperbaikinya.
2. UML memberikan visualisasi dan pemodelan dengan lebih jelas dari sistem yang kompleks.
3. Teknik rekayasa *software* adalah teknik visual programming dibutuhkan untuk meningkatkan kualitas dan mengurangi biaya dan waktu.
4. Membuat suatu himpunan semantik dan notasi yang mampu menangani kerumitan arsitektural dalam semua ruang lingkup (*Suhendar dkk, 2002*).

Visual Modelling adalah proses menggambarkan cetak biru suatu sistem secara grafis, terdiri dari komponen-komponen, *interface* dan koneksi-koneksi yang ada dalam sistem tersebut, agar mudah dipahami dan dikomunikasikan (*Suhendar dkk, 2002*).

Suatu model dari *software* yang dibangun memiliki berbagai cara pandang (*view*), dan dibagi menjadi :

1. *Use case view*

   Membantu dalam pemahaman dan penggunaan sistem yang dimodelkan. *View* ini melihat pada bagaimana *actor* dan *use-case* berinteraksi.

2. *Logical view*

   Mengarahkan pada persyaratan (*requirements*) fungsional sistem. *View* ini melihat pada kelas-kelas dan hubungan antar kelas tersebut.

3. *Component view*

   Mengarah pada pengaturan software. *View* ini mengandung informasi mengenai komponen-komponen *software*, komponen tereksekusi (*executeable*) dan *library* untuk sistem yang kita modelkan.

4. *Deployment view*

   Memperlihatkan pemetaan setiap proses ke dalam *hardware*. *View* ini bermanfaat pada model suatu sistem yang diterapkan dalam lingkungan arsitektur yang terdistribusi dimana kita menerapkan aplikasi dan *server* pada lokasi yang berbeda.

Beberapa diagram UML dalam penggambaran *visual modelling*:

1. *Class Diagram*

   Digunakan untuk menampilkan hubungan antar kelas-kelas dan informasi mengenai suatu kelas. *Class Diagram* adalah dasar dari *Component Diagram* dan *Deployment Diagram*

2. *Use-case Diagram*

   Digunakan untuk memperlihatkan sistem secara garis besar *high level view*, terutama dilihat dari perspektif pengguna. Diagram ini menunjukan fungsi-fungsi sistem dan bagaimana sistem berinteraksi dengan dunia luar. Pada tahap analisis, *use-case* diagram digunakan untuk menjabarkan perilaku sistem jika rancangan diimplementasikan.

3. *Sequence Diagram*

   Digunakan untuk menggambarkan interaksi objek-objek berdasarkan urut-urutan waktu (*time sequence*). *Sequence Diagram* biasanya digunakan

bersamaan dengan *use-case* diagram untuk menunjukkan langkah-langkah yang akan dilakukan dalam melakukan salah satu proses yang ada dalam *Use-Case Diagram*

4. *Collaboration Diagram*

   Digunakan untuk menggambarkan interaksi atau hubungan struktural antar objek-objek dalam model. *Collaboration Diagram* dapat dipakai untuk mengekspresikan jalur-jalur keputusan dari sistem.

5. *Activity Diagram*

   Digunakan untuk memodelkan alur dari proses bisnis dan urut-urutan aktifitas dalam sebuah proses. Diagram ini mirip dengan *flowchart*. Gunakan diagram ini di awal pemodelan suatu proses untuk membantu memahami proses secara keseluruhan. Diagram ini juga berguna untuk menggambarkan bagaimana proses-proses yang paralel atau untuk menggambarkan bagaimana proses-proses dalam beberapa *use-case* berinteraksi.

6. *Component Diagram*

   Digunakan untuk menunjukkan gambaran fisik dari model, menunjukan organisasi dan keterkaitan antar komponen-komponen *software,* termasuk source code, *binary code* dan *executeable component.*

7. *Deployment Diagram*

   Setiap model memiliki sebuah *deployment* diagram yang menunjukan pemetaan dari proses-proses ke *hardware*.

8. *Statechart Diagram*

   Digunakan untuk memodelkan perilaku dinamik dari sebuah kelas (atau objek). Diagram ini menunjukan urut-urutan state yang dilalui sebuah objek, *event* yang menyebabkan perpindahan dari satu *state* ke *state* lain dan aksi-aksi yang dilakukan pada sebuah *state*. Diagram ini biasanya digunakan untuk memodelkan tahap-tahap dari kehidupan sebuah objek (*Suhendar dkk, 2002*).

Dalam UML, hubungan atau relasi antar *class* atau objek digambarkan dalam *objek diagram*. Relasi antar *class* tersebut dapat berupa :

1. Asosiasi (*association*) adalah hubungan koneksi semantik (*semantic connenction*) antar class. Misalnya suatu *class* penulis dihubungkan dengan *class* buku dengan hubungan asosiasi membuat, atau suatu *class plant* dihubungkan dengan *class* pengontrol dengan hubungan mempunyai. Hubungan agregasi merupakan kasus khusus hubungan asosiasi dimana suatu *class* mempunyai hubungan memiliki (*contains*) pada suatu *class* lain, misalnya class plant memiliki *class* tangki.
2. Generalisasi (*generalizaton*) yaitu hubungan antar elemen yang lebih general dengan elemen yang lebih spesifik yang masih memiliki sifat induknya hanya saja memiliki informasi lain.
3. Ketergantungan (*dependency*) yaitu hubungan antar elemen *independent* dan elemen *dependent*. Perubahan pada elemen *independent* akan mengakibatkan perubahan pula pada elemen yang *dependent* terhadapnya.

4. *Refinement* yaitu hubungan antara dua deskripsi sesuatu tetapi memiliki level perbedaan abstrak (*Suhendar dkk, 2002*).

2.5 Java

Bahasa pemrograman Java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992.. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995. Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

2.5.1 Karakteristik Dan Kelebihan Java

Java memiliki beberapa karakteristik yang menjadikan Java memiliki kelebihan dibanding bahasa pemrograman lain, yakni (id.wikipedia.org, 2009) :

1. *Multiplatform.*

   Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* atau sistem operasi komputer, sesuai dengan prinsip "*tulis sekali, jalankan di*

mana saja”. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin (*bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris.

2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek)

   Semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *Object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.

3. Perpustakaan Kelas Yang Lengkap

   Java terkenal dengan kelengkapan *library* (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java

yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

4. Bergaya C++

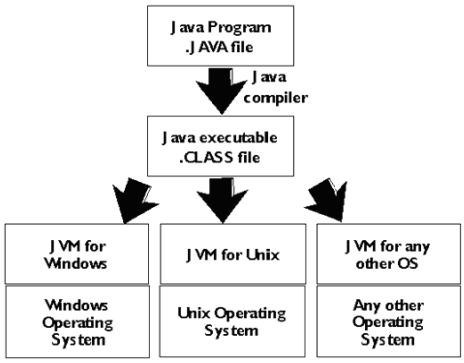
   Memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.

5. Pengumpulan sampah (*Garbage*)otomatis

   Memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.5.2 Cara Kerja Java

Java merupakan bahasa pemrograman *compiler* dan juga *interpreter* yang dapat dijalankan pada berbagai *platform*. Java *compiler* melakukan kompilasi pada *source code* menjadi *Java bytecodes*. *Java bytecode* merupakan instruksi mesin yang tidak spesifik terhadap prosesor komputer dan akan dijalankan pada *platform* menggunakan *Java Virtual Machine* (JVM) yang biasa disebut *bytecodes interpreter* atau *Java run-time interpreter*. (*Wahana Komputer, 2005*)



Gambar 2. 15 Lingkungan Java (Wahana Komputer, 2005)

2.5.3 Java 2D API

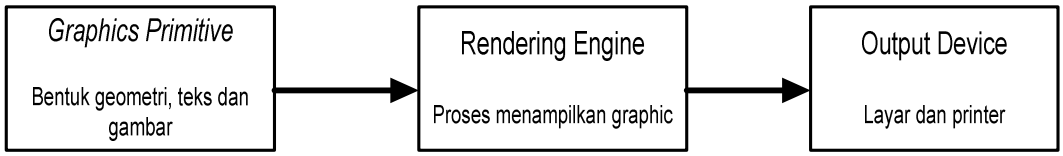
Peta Kontur merupakan gambaran warna dua dimensi.. Java menyediakan serangkaian *class* yang dapat digunakan untuk menghasilkan grafik-grafik. *Class-class* ini dikenal dengan Java 2D *Application Programing Interface* (2D API) (*Leonardo, 2003*).

Java 2D API ini merupakan bagian dari *core* java 2 platform, yang memperluas kemampuan AWT (*Abstract Windowing Toolkit*) untuk menangani grafik, teks dan gambar. Java 2D API mendukung definisi warna dan komposisi, *hit detection* pada sembarang bentuk geometri dan teks, model *rendering* yang *uniform* untuk printer dan *display device*. Jika digunakan bersama dengan Java *Media Framework* dan Java Media API lainnya, Java 2D dapat digunakan untuk membuat animasi dan menampilkan presentasi multimedia. Java Animation dan Java Media Framework API bergantung pada Java 2D API untuk dukungan *rendering* (*Leonardo, 2003*).

Untuk menampilkan objek kontur, perlu ditetapkan sebuah konteks grafik dan memanggil metode untuk melakukan *rendering*. *Rendering* adalah proses

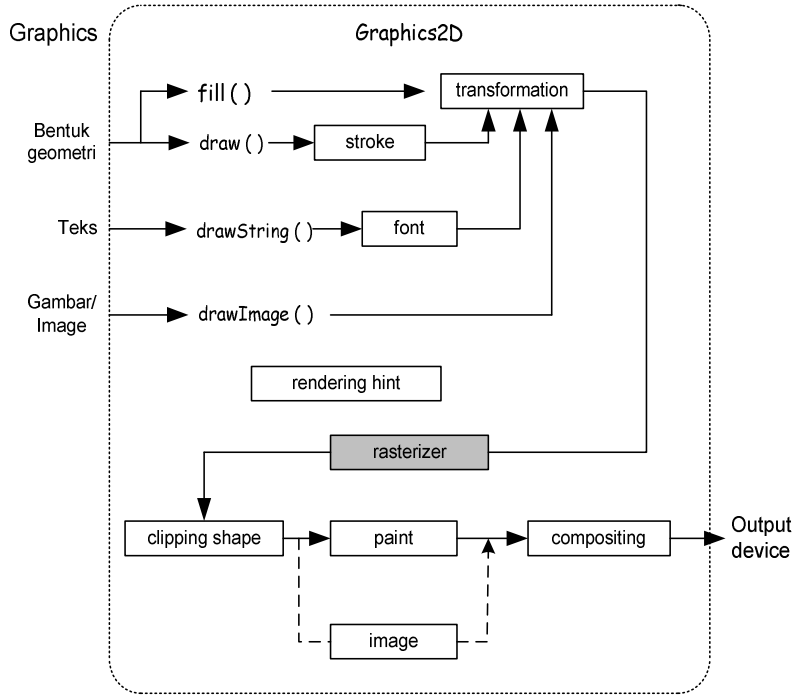
mengambil sekumpulan bentuk geometri, teks dan gambar, serta menentukan warna yang akan muncul pada layar dan printer. Bentuk geometri, teks dan gambar disebut dengan *graphic primitive*, layar dan printer disebut dengan *output device* atau *rendering device*. Jadi, *rendering* adalah proses menampilkan *graphics primitive* ke *output device*. Proses ini dilakukan oleh *rendering engine*.

Gambar di bawah ini memperlihatkan proses *rendering* secara garis besar (*Leonardo, 2003*).



Gambar 2. 16 Proses rendering secara garis besar

Ada beberapa teknik yang berbeda yang dapat digunakan untuk melakukan *render* pada *graphics primitive*. *Rendering hint* memberitahukan ke *Graphic 2D* teknik yang mana yang akan digunakan. Gambar berikut ini memperlihatkan proses rendering secara detail *(Leonardo, 2003*).



Gambar 2. 17 Proses rendering dalam Graphic2D secara detail.

Gambar ditampung di dalam sesuatu yang disebut dengan *raster*. *Raster* dipresentasikan oleh *class java.awt.image.raster*. *Raster* terdiri atas dua bagian, yaitu *buffer data* dan model sampel. *Buffer data* berisi data mentah (nilai sampel), sedangkan model sampel mendeskripsikan bagaimana data tersebut diorganisasi.

Ruang warna adalah sekumpulan warna-warna yang dapat ditampilkan oleh *device* tertentu. Sebagai contoh monitor RGB, memiliki jangkauan warna-warna yang dapat ditampilkan berdasarkan intensitas cahaya merah, hijau dan biru yang ada padanya. Sebuah monitor RGB yang berbeda akan mempunyai ruang warna yang berbeda, karena adanya variasi cahaya merah, hijau dan biru yang disebabkan komponen elektronik yang berbeda. Kedua monitor mempunyai tipe ruang warna yang sama, yaitu RGB, tetapi masing-masing mempunyai ruang warna yang *device-dependent.* Hal yang sama, beberapa printer menggunakan

kombinasi tinta cyan, magenta, kuning (yellow), dan hitam (black) untuk menghasilkan warna. Printer tipe ini mempunyai ruang warna bertipe CMYK, tetapi setiap printer mendefinisikan ruang warna CMYK masing-masing.

Beberapa jenis ruang warna yang lain :

1. CIEXYZ

   Pada tahun 1931, CIE (*Commission Internationale de l'Eclairage*) mendesain ruang warna absolut. Mereka mendefinisikan tiga warna primer yang disebut X, Y, dan Z. Semua warna dapat dihasilkan dengan hanya menggunakan nilai positif dari X, Y, dan Z.

   Pada umumnya, tidak nyaman bagi device untuk berhubungan dengan CIEXYZ. Sebuah monitor menghasilkan warna-warna dengan mencampurkan cahaya merah, hijau, dan biru, sehingga sistem RGB sangat alamiah. Dengan cara yang sama, kebanyakan printer bekerja dengan mencampurkan tinta berwarna cyan, magenta, kuning, dan hitam.

   Walaupun demikian, sistem CIEXYZ berguna dengan dua alasan:

   a. CIEXYZ adalah absolut. Ia tidak bergantung pada sembarang device.
   b. Setiap warna dapat dipresentasikan dengan menggunakan CIEXYZ

2. sRGB

   Ruang warna absolut yang lebih baru adalah sRGB (*standard* RGB). Ruang warna ini menggunakan warna primer, hijau, dan biru, tetapi mendefinisikan mereka secara absolut. Seperti CIEXYZ, sRGB adalah standar, tidak tergantung pada *device* (*device-independent*). Ia lebih mudah digunakan dibandingkan CIEXYZ karena menggunakan tipe ruang warna RGB yang

telah biasa dikenal, tetapi mempunyai kemampuan yang lebih rendah dibandingkan CIEXYZ karena tidak mampu menampilkan semua warna yang bisa dilihat. sRGB adalah ruang warna *default* pada java 2D.

Class color juga mendefinisikan beberapa konstanta yang menyimpan warna-warna yang sering digunakan. Seperti tabel warna di bawah ini :

Tabel 2. 2 Warna *predefined* dalam *color*

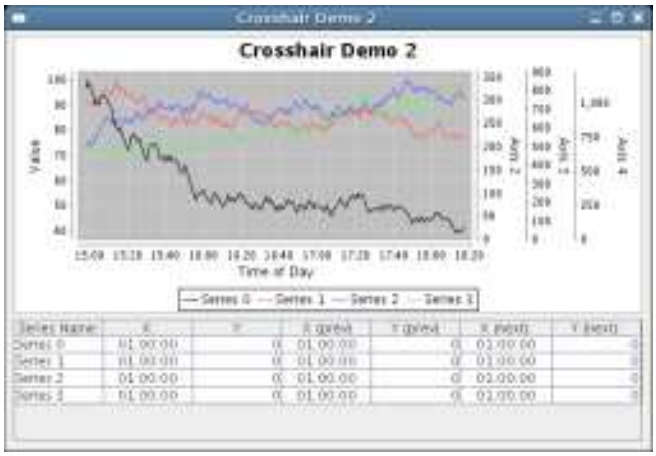
Konstanta	**ekivalen**
Color.black	*new color* (0, 0, 0)
Color.blue	*new color* (0, 0, 255)
Color.cyan	*new color* (0, 255, 255)
color.darkgray	*new color* (64, 64, 64)
Color.gray	*new color* (128 , 128, 128)
Color.green	*new color* (0, 255, 0)
Color.lightgray	*new color* (192, 192, 192)
Coor.magenta	*new color* (255, 0, 255)
Color.orange	*new color* (255, 200, 0)
Color.pink	*new color* (255, 175, 175)
Color.red	*new color* (255, 0, 0)
Color.white	*new color* (255, 255, 255)
Color.yellow	*new color* (255, 255, 0)

2.5.4 JFreeChart

JFreeChart proyek yang didirikan delapan tahun lalu, pada bulan Februari 2000, oleh David Gilbert. Hari ini, JFreeChart digunakan oleh sekitar 40.000-50.000 pengembang. Proyek ini akan terus dikelola oleh Gilbert, dengan sumbangan dari masyarakat yang beragam dari para pengembang.

JFreeChart adalah Java *Library* grafik yang akan memudahkan bagi pengembang untuk menampilkan kualitas grafik profesional dalam aplikasi mereka. JFreeChart memiliki fitur untuk mengatur :

1. Memliki banyak *library* dan dokumentasi API, mendukung berbagai jenis grafik bagan (*chart*).
2. Desain yang fleksibel dan mudah digunakan, baik dari sisi server dan aplikasi client-side untuk pemrograman java berbasis web.
3. dukungan untuk berbagai jenis output, termasuk komponen swing, file gambar (termasuk PNG dan JPEG), dan format file grafis vektor (termasuk PDF, EPS dan SVG).
4. JFreeChart adalah "*open source*" atau, lebih khusus, perangkat lunak bebas untuk dipakai dan dikembangkan.. Hal ini didistribusikan menurut ketentuan Lesser General Public Licence (LGPL), yang memungkinkan penggunaan yang eksklusif pada suatu aplikasi.



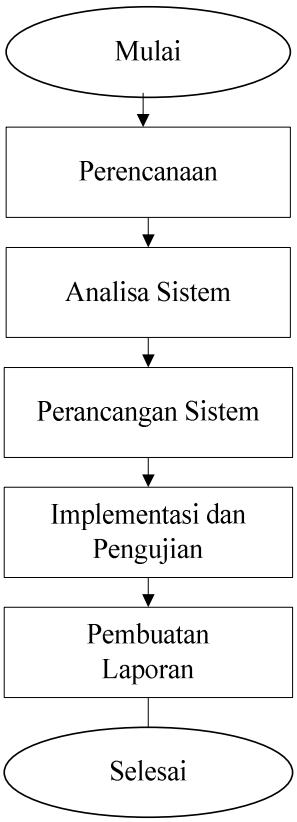
Gambar 2. 18 Contoh Model Tampilan JFreeChart

JFreeChart mendukung grafik pie (2D dan 3D), bar charts (vertikal dan horisontal, dan reguler), grafik baris, *contour plot*, grafik harian, termometer, cepat, dan banyak lagi. JFreeChart dapat digunakan dalam aplikasi, applet, servlet dan JSP (*Java Server Pages*).

BAB III

METODE PENELITIAN

Rekayasa perangkat lunak memiliki berbagai proses yang bertujuan agar menjadikan perangkat lunak yang dibuat sesuai dengan yang diinginkan. Rekayasa merupakan tahapan proses analisis, desain, konstruksi, verifikasi, dan manajemen kesatuan dalam membangun suatu aplikasi peta kontur dengan menggunakan algoritma kriging untuk memecahkan masalah dalam analisa perminyakan.



Gambar 3. 1 Diagram Alir Pembuatan Aplikasi Oil System

Tahap yang harus dilalui dalam pengembangan system adalah sebagai berikut:

1. Tahap Perencanaan

   Tahapan Perencanaan yang akan dilakukan dalam melakukan penelitian, yang meliputi pengumpulan data dan analisa kebutuhan untuk membuat perencanaan terhadap aplikasi peta kontur yang akan dibuat sebagai dasar untuk melakukan pengembangan aplikasi Oil System v1.0 sebelumnya.

   Cara mengumpulkan data dalam penelitian ini digunakan beberapa teknik sebagai berikut :

   a. *Field Penelitian Research* (penelitian lapangan)

      Penelitian yang dilakukan yaitu dengan cara Observasi langsung melihat kinerja aplikasi lain yang menyerupai dengan aplikasi yang akan dibuat. Contoh : Aplikasi GS+ yang dibuat oleh GammaDesign.

   b. *Library Research* (Penelitian Kepustakaan)

      Penelitian kepustakaan dengan membandingkan informasi yang telah di dapat dari beberapa sumber buku maupun internet bertujuan untuk membandingkan antar informasi mengenai peta kontur dan solusi dengan algoritma kriging, sehingga penyelesaian dan pembahasan lebih komplit dan benar.

2. Tahap Analisis

   Merupakan tahap yang paling penting dalam pembuatan aplikasi peta kontur.

Hal-hal yang dilakukan dalam analisa sistem sebagai berikut :

a. mengidentifikasi masalah-masalah yang ada.

   Identifikasi masalah dalam memprediksi titik-titik kontur sumur minyak yang tidak diketahui agar terbentuk peta kontur yang akurat pada suatu lapangan minyak yang dapat dimanfaatkan dalam analisa perminyakan.

b. menemukan solusi yang tepat dalam permasalahan yang ada.

   Menemukan solusi dalam memprediksi titik-titik kontur yang tidak diketahui dengan menggunakan persamaan algoritma kriging sehingga peta kontur yang dibentuk memiliki tampilan grafik yang lebih baik.

c. mengetahui kerja dari sistem yang ada

   Dimulai dari proses pengambilan data sumur sebagai data lokasi titik kontur dan data produksi sumur sebagai data besar kandungan minyak, prediksi titik kontur lain yang tidak diketahui dengan memanfaatkan data lokasi titik kontur dan data kandungan produksi minyak dengan melakukan perhitungan dengan persamaan kriging, dan terbentuknya peta kontur kandungan minyak pada suatu lapangan minyak.

d. mendefinisikan kebutuhan pengguna.

   Mendefinisikan layanan sistem yang dapat digunakan oleh pengguna, meliputi model peta kontur 2 Dimensi (2D) dan grafik peta kontur dapat disimpan menjadi file gambar berekstensi *.png serta dapat dicetak.

3. Tahap Perancangan

   Dalam tahap ini terdapatkan gambaran yang jelas tentang sistem yang akan dibuat. Sistem yang akan dibuat dirancang berdasarkan fakta dari analisa

sistem yang telah dilakukan, meliputi pembuatan diagram *Unified Modelling Language (*UML*),* perancangan antar muka (*design interface*) dan melakukan pembuatan program menggunakan kode program Java dan basisdata MySQL.

4. Tahap Implementasi dan Pengujian

   Tahap ini merupakan tahap dimana suatu sistem siap untuk di coba. Dari percobaan ini diharapkan dapat diketahui kinerja dan performa sistem yang harus sesuai dengan yang diinginkan dan sesuai dengan hasil yang ingin dicapai melalui pengujian sistem yang telah dibuat apakah masih terdapat error sistem atau tidak.

5. Tahap Pembuatan Laporan

   Merupakan tahap akhir dalam sebuah pengembangan perangkat lunak dengan melakukan dokumentasi terhadap sistem yang telah dibuat dengan menggunakan aturan atau prosedur yang telah ditetapkan dalam penulisan laporan tugas akhir.

BAB IV

ANALISA DAN PERANCANGAN

4.1 Analisa Sistem

Informasi perkembangan dalam pengelolaan sumur minyak sangat diperlukan dalam suatu perusahaan minyak agar pengelolaan sumur minyak dapat dilakukan dengan efektif dan efisien. Analisa perminyakan dilakukan dengan memanfaatkan data-data beberapa sumur minyak dalam berapa periode waktu agar diketahui perkembangan dari sumur tersebut. Untuk memanfaatkan data sumur minyak menjadi informasi yang lebih baik dan berguna dalam analisa perminyakan, maka data sumur tersebut dapat digambarkan dalam bentuk grafik, baik berupa *chart*, peta 2D, maupun peta 3D.

Sistem yang menjadi tugas akhir sebelumnya hanya analisa berupa grafik *chart*. Dalam pengembangan selanjutnya dilakukan dengan analisa peta kontur 2D. Peta *countur* 2D dapat mempermudah analisa daerah-daerah maupun sumur-sumur yang memiliki nilai yang tinggi sebagai daerah yang berproduksi tinggi, daerah yang berproduksi rendah, daerah yang hampir habis minyaknya, daerah yang berproduksi tinggi tapi minyaknya tidak habis. Beberapa aplikasi dapat menggambarkan peta kontur 2D dari data sumur minyak, namun kebanyakan aplikasi tersebut memproses data sumur minyak berbentuk teks bukan data yang tersimpan dalam DBMS (*Database Management System*) dan format data teks yang digunakan juga berbeda-beda. Aplikasi yang menggambarkan peta kontur yang warna dan garis kontur yang halus harganya mahal sedangan kebanyakan

aplikasi yang tidak berbayar dan *opensource* memiliki kelemahan dalam membentuk peta countur, dimana peta *countur* yang dibentuk kurang baik sebagai bahan analisa perminyakan.

Pembuatan aplikasi yang akan diberi nama “Oil System v1.1” ini akan dimulai dengan mengidentifikasi kebutuhan-kebutuhan perangkat lunak (*software requirements*). Yang merupakan bagian terpenting untuk perilaku aplikasi nantinya dalam menyelesaikan suatu permasalahan.

4.1.1 Analisa Kebutuhan Fungsional

Menggambarkan karakteristik atau perilaku aplikasi untuk menyelesaikan suatu masalah yang di-*input* pengguna. Dan lebih lanjutnya akan dijabarkan ke dalam kebutuhan pengguna (*user requirement*), dan kebutuhan pemakaian grafis (GUI – *Graphical User Interface*) yang menginterpretasikan kebutuhan pengguna.

4.1.1.1 Proses Pembentukan Peta *Contour* dengan Algoritma Kriging

Konsep dari pemetaan kontur sumur minyak dengan menggunakan algoritma kriging dapat kita lihat melalui analisa contoh berikut dimana nilai di titik u yang terletak pada titik x = 2000 meter dan y = 4700 meter akan dicari nilai Z nya berdasarkan 6 titik data yang berada pada suatu lokasi di suatu lapangan minyak. Langkah – langkah yang dilakukan sebagai berikut :

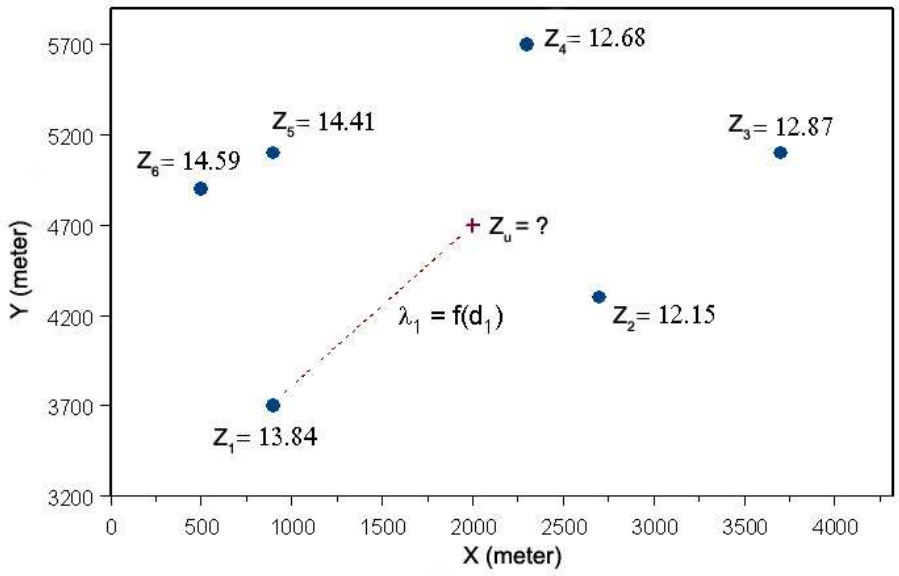
1. Menentukan jarak masing-masing titik terhadap titik yang dicari berdasarkan persamaan 2.1.
2. Menentukan bentuk semivariogram spherical dengan melakukan perhitungan persamaan 2.10

3. Mencari nilai Z dan menggambarkannya dalam peta kontur berdasarkan persamaan 2.15
4. Membentuk peta kontur dengan mengambil 50 titik sample lain.

Tabel 4. 1 Data Sumur

Sumur	**Lokasi X**	**Lokasi Y**	**Produksi minyak (Z)**
1	900	3700	13,84
2	2100	4300	12,15
3	3700	5100	12,87
4	2300	5700	12,68
5	900	5100	14,41
6	500	4900	14,59
U	2000	4700	(tidak diketahui)

Dari tabel diatas dapat digambarkan bentuk masalah dengan pemetaan sumbu x dan y.



Gambar 4. 1 Gambaran data yang diketahui

Dari contoh data diatas untuk memperkirakan nilai dititik u dengan melakukan perhitungan persamaan kriging (2.10) $\sum \omega_i Z_i$, dengan menghitung bobot (ω_i) dengan menghitung jarak (d) dari titik u ketitik lain.

Langkah pertama yang dilakukan adalah dengan menghitung jarak masing masing titik terhadap titik u berdasarkan persamaan 2.1.

$$h_{iu} = \sqrt{(x_u - x_i)^2 + (y_u + y_i)^2}$$

$$h_{1u} = \sqrt{(2000 - 900)^2 + (4700 - 3700)^2}$$

$$= \sqrt{2211169}$$

= 1487 m

$$h_{2u} = \sqrt{(2000 - 2700)^2 + (4700 - 4300)^2}$$

$$= \sqrt{649636}$$

= 806 m

$$h_{3u} = \sqrt{(2000 - 3700)^2 + (4700 - 5100)^2}$$

$$= \sqrt{3048516}$$

= 1746 m

Dengan melakukan perhitung hingga h_{4u}, h_{5u},dan h_{6u}, maka akan menghasilkan blok lokasi terhadap sumbu x dan y.

	$(x,y)_1$	$(x,y)_2$	$(x,y)_3$	$(x,y)_4$	$(x,y)_5$	$(x,y)_6$
$(x,y)_u$	1487	806	1746	1044	1170	1513

Setelah jarak didapat maka kita dapat menghitung persamaan semivariogram spherical $\gamma(h)$ 2.6 dengan menetapkan effect nugget (C_0) = 0, *sill* (C) = 0.78, dan *range* (a) = 4141 meter.

$$\gamma(h_{iu}) = C_0 + C\left(\frac{3h_{iu}}{2a} - \frac{1h_{iu}^3}{2a^3}\right)$$

$\gamma_{1u}(1487)$ = 0.78(1.5(1487/4141) – 0.5(1487/4141)^3)

= 0.38

$\gamma_{2u}(806)$ = 0.78(1.5(806/4141) – 0.5(806/4141)^3)

= 0.56

$\gamma_{3u}(1746)$ = 0.78(1.5(1746/4141) – 0.5(1746/4141)^3)

= 0.32

$\gamma_{4u}(1044)$ = 0.78(1.5(1044/4141) – 0.5(1044/4141)^3)

= 0.49

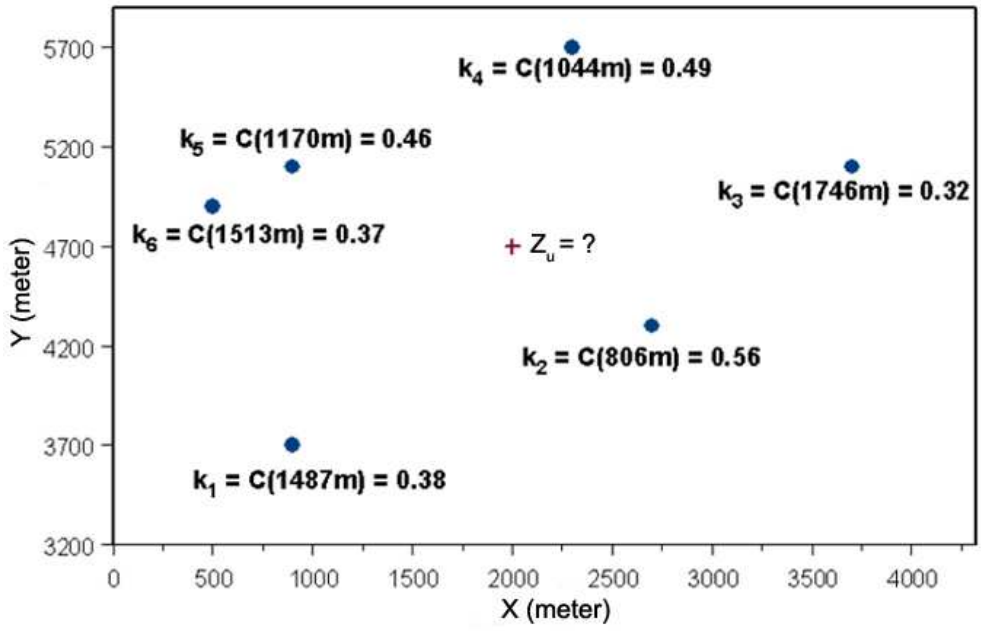
$\gamma_{5u}(1170)$ = 0.78(1.5(1170/4141) – 0.5(1170/4141)^3)

= 0.46

$\gamma_{6u}(1513)$ = 0.78(1.5(1513/4141) – 0.5(1513/4141)^3)

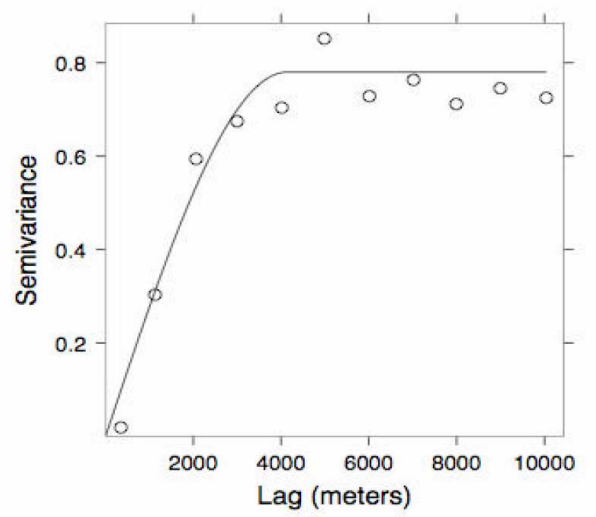
= 0.37

Hasil perhitungan ini dapat digambarkan sebagai berikut :



Gambar 4. 2 Hasil Perhitungan Semivariogram

Dengan menghitung sembarangan titik sampel lain, maka akan membentuk grafik kurva semivariogram spherical.



Gambar 4. 3 Semivariogram Spherical dari 6 titik sampel

Untuk melalukan proses perhitungan kriging, kita perlu menghitung semivariogram antar titik-titik yang diketahui dengan terlebih dahulu menghitung jarak antar titik-titik seperti langkah sebelumnya.

Tabel 4. 2 Jarak antar sumur

Jarak	**Sumur 1**	**Sumur 2**	**Sumur 3**	**Sumur 4**	**Sumur 5**	**Sumur 6**
Sumur 1	0	1897	3130	2441	1400	1265
Sumur 2	1897	0	1281	1456	1970	2280
Sumur 3	3130	1281	0	1523	2800	3206
Sumur 4	2441	1456	1523	0	1523	1970
Sumur 5	1400	1970	2800	1523	0	447
Sumur 6	1265	2280	3206	1970	447	0

Setelah semua jarak didapat lalu kita memulai proses pencarian bobot (ω) berdasarkan persamaan 2.16 yang berbentuk matrik.

$$\begin{pmatrix} \gamma(h_{11}) & \gamma(h_{12}) & \gamma(h_{13}) & \gamma(h_{14}) & \gamma(h_{15}) & \gamma(h_{16}) & 1 \\ \gamma(h_{21}) & \gamma(h_{22}) & \gamma(h_{23}) & \gamma(h_{24}) & \gamma(h_{25}) & \gamma(h_{26}) & 1 \\ \gamma(h_{31}) & \gamma(h_{32}) & \gamma(h_{33}) & \gamma(h_{34}) & \gamma(h_{35}) & \gamma(h_{36}) & 1 \\ \gamma(h_{41}) & \gamma(h_{42}) & \gamma(h_{43}) & \gamma(h_{44}) & \gamma(h_{45}) & \gamma(h_{46}) & 1 \\ \gamma(h_{51}) & \gamma(h_{52}) & \gamma(h_{53}) & \gamma(h_{54}) & \gamma(h_{55}) & \gamma(h_{56}) & 1 \\ \gamma(h_{61}) & \gamma(h_{62}) & \gamma(h_{63}) & \gamma(h_{64}) & \gamma(h_{65}) & \gamma(h_{66}) & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \\ \omega_6 \\ \lambda \end{pmatrix} = \begin{pmatrix} \gamma(h_{1p}) \\ \gamma(h_{2p}) \\ \gamma(h_{3p}) \\ \gamma(h_{4p}) \\ \gamma(h_{5p}) \\ \gamma(h_{6p}) \\ 1 \end{pmatrix}$$

$$\begin{bmatrix} 0.78 & 0.28 & 0.06 & 0.17 & 0.40 & 0.43 & 1.00 \\ 0.28 & 0.78 & 0.43 & 0.39 & 0.27 & 0.20 & 1.00 \\ 0.06 & 0.43 & 0.78 & 0.37 & 0.11 & 0.06 & 1.00 \\ 0.17 & 0.39 & 0.37 & 0.78 & 0.37 & 0.27 & 1.00 \\ 0.40 & 0.27 & 0.11 & 0.37 & 0.78 & 0.65 & 1.00 \\ 0.43 & 0.20 & 0.06 & 0.27 & 0.65 & 0.78 & 1.00 \\ 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.00 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \\ \omega_6 \\ \lambda \end{bmatrix} = \begin{bmatrix} 0.38 \\ 0.56 \\ 0.32 \\ 0.49 \\ 0.46 \\ 0.37 \\ 1.00 \end{bmatrix}$$

Dengan melakukan inverse (membalikan) matrik kita akan mendapatkan nilai masing-masing bobot.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \\ \omega_6 \\ \lambda \end{bmatrix} = \mathbf{K}^{-1}\mathbf{k} = \begin{bmatrix} 0.1274 \\ 0.4515 \\ -0.0463 \\ 0.2595 \\ 0.2528 \\ -0.0448 \\ 0.0288 \end{bmatrix}$$

Berdasarkan persamaan 2.13 kita dapat menghitung nilai perkiraan (Z) setelah nilai bobot (w) diketahui.

$$\hat{Z}_u = \omega_1 Z_1 + \omega_2 Z_2 + \omega_3 Z_3 + \omega_4 Z_4 + \omega_5 Z_5 + \omega_6 Z_6$$

$\hat{Z}_u$ = 0,1274(13,84) + 0,4515(12,15) - 0.0463(12,87) + 0,2595 (12,68) +

0,2528(14,41) - 0,00448 (14,59)

= 1,76 + 5,48 – 0,59 + 3,29 + 3,64 – 0,065

= 13,15

Sedangkan unutk perkiraan error berdasarkan persamaan 2.17 adalah sebagai berikut :

$$s_E^2 = \omega_1\gamma(h_{1u}) + \omega_2\gamma(h_{2u}) + \omega_3\gamma(h_{3u}) + \omega_4\gamma(h_{4u}) + \omega_5\gamma(h_{5u}) + \omega_6\gamma(h_{6u}) + \lambda$$

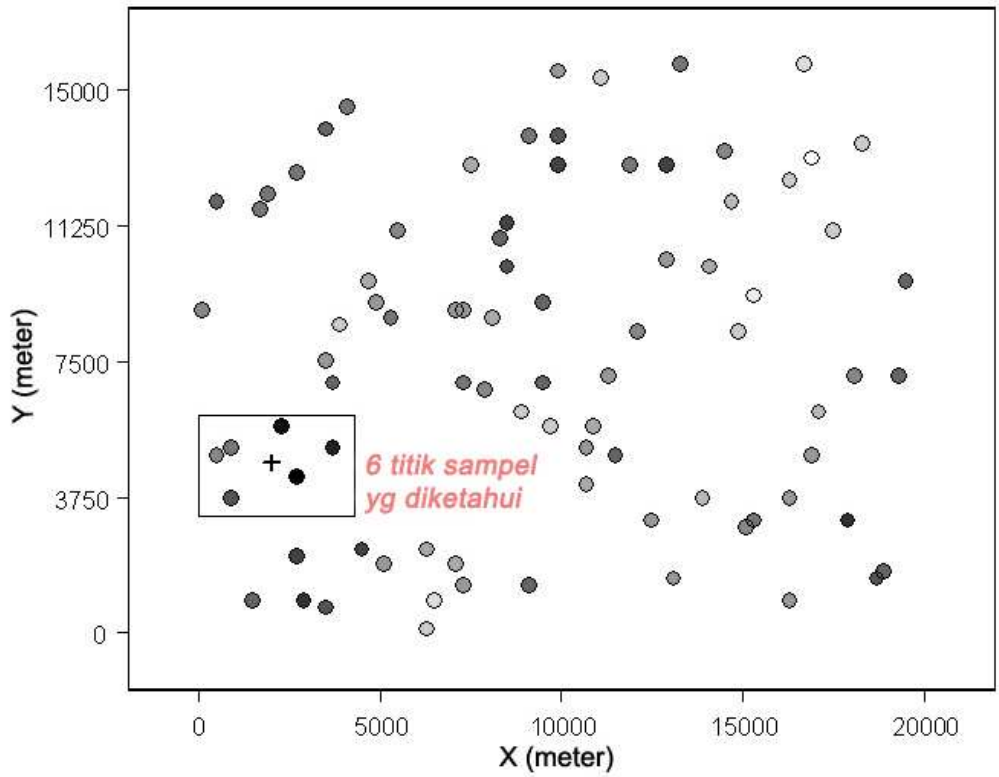
s_E^2 = 0,1274(0,38) + 0,4515(0.56) - 0.0463(0,32) + 0,2595 (0,49) +

0,2528(0.46) - 0,00448 (0,37) + 0,0288

= 0.048412+0.25284-0.014816+0.127155+0.116288-0.0016576+0,0288

= 0.5570214

Dengan mengambil 50 titik sampel sembarangan terdekat dari 6 titik yang diketahui maka kita akan mendapatkan 50 titik baru dengan membatasi hanya 100x80 grid dengan jarak 100 meter dan lokasi X hingga 20 Km dan lokasi Y hingga 15 km menggunakan semivariogram spherical, maka akan terbentuk titik-titik kontur baru seperti gambar berikut ini :

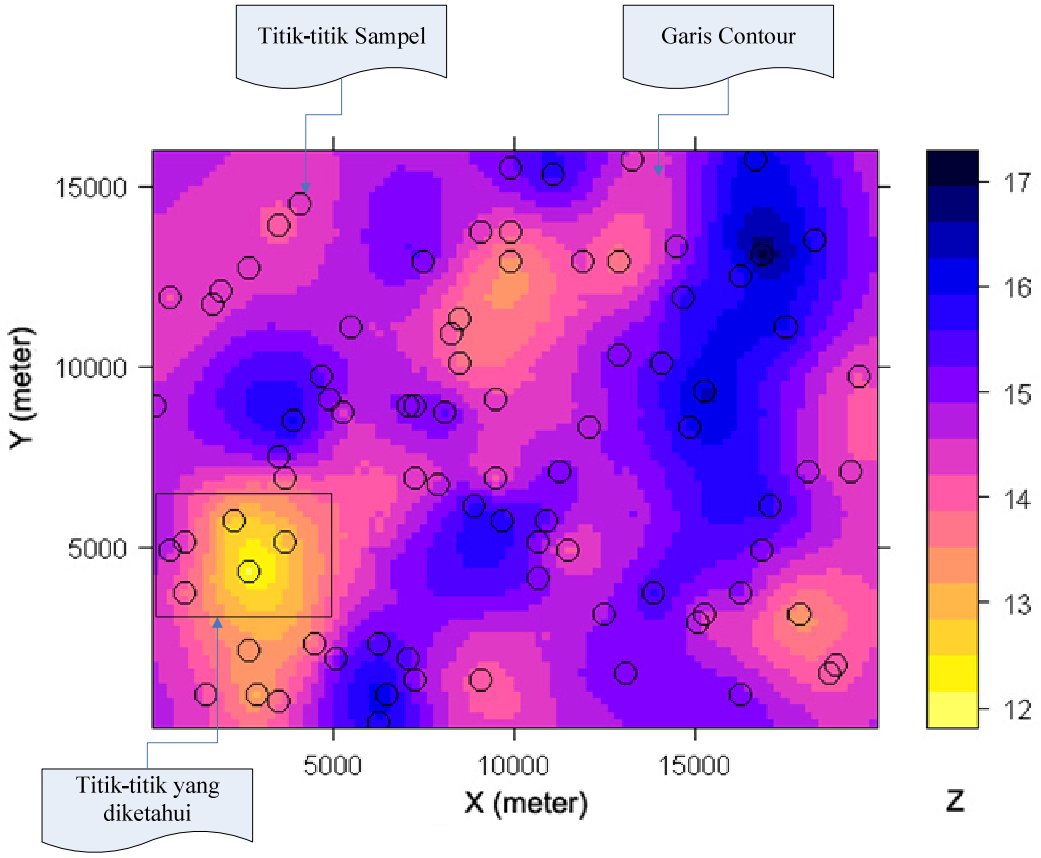


Gambar 4. 4 Hasil 50 titik sampel baru

Dari gambar diatas dapat kita lihat 6 titik sampel yg diketahui yang ditandai dengan persegi dan titik-titik sampel lain yang dimulai dari 6 titik sampel hingga titik-titik lain yang saling berdekatan sehingga terbentuk titik-titik baru dengan mengikuti langkah-langkah yang dilakukan untuk memprediksi nilai perkiraan kandungan minyak pada suatu titik.

Setelah terbentuk titik-titik kontur seperti gambar sebelumnya, kita perlu menambahkan tingkatan warna yang menunjukan tingkat kandungan minyak agar terbentuk sebuah peta kontur. Untuk menambahkan tingkatan warna kita harus membentuk garis-garis kontur yang membatasi luas cakupan wilayah kandungan minyak dengan kembali melakukan perhitungan kriging dengan menghitung

kandungan minyak ditiap sumbu X dan Y sehingga terbentuk titik-titik kontur yang berdekatan membentuk sebuah garis kontur.



Gambar 4. 5 Peta Kontur Sumur Minyak pada suatu lapangan minyak

4.1.1.2 Kebutuhan Pengguna (*User Requirements*)

Analisa fungsional berdasarkan kebutuhan pengguna (*user requirements*) sebagai berikut :

1. Pengguna dapat melakukan pengolahan data.

Tabel 4. 3 Kebutuhan Pengguna untuk pengolahan data peta

Kandidat *use-case*	**Melakukan Pengolahan Data**

Perilaku user yang mungkin terjadi	Memasukan data koordinat dan nilai variabel yang dihitung
Tingkat kebutuhan	Primer
Catatan	diimplementasikan

2. Pengguna dapat melihat peta kontur lapangan minyak.

Tabel 4. 4 Kebutuhan Pengguna untuk Membuka Peta Kontur

Kandidat *use-case*	**Membuat peta kontur dengan algoritma kriging**
Perilaku user yang mungkin terjadi	Membuka peta kontur
Tingkat kebutuhan	Primer
Catatan	diimplementasikan

4.1.1.3 Kebutuhan GUI (Graphical User Interface Requirements)

Menggambarkan kebutuhan pemakaian grafis (GUI – *Graphical User Interface*) yang menginterpretasikan kebutuhan pengguna.

1. Dibutuhkan sebuah layar input data untuk melakukan pengolahan data peta Kontur.

   Sistem aplikasi akan memanggil modul untuk melakukan pengolahan data yang dinamakan ’Plot Workshhet’ sebagai *interface* dengan pengguna.

2. Dibutuhkan sebuah layar pensettingan variabel-variabel dalam algoritma kriging untuk membentuk suatu peta kontur yang sesuai dengan yang diinginkan pengguna.

   Sistem aplikasi akan memanggil modul input untuk pensettingan variabel kriging yang dinamakan ’*Contour Plot*’ sebagai *interface* dengan pengguna

3. Dibutuhkan sebuah layar peta kontur untuk memvisualisasikan panel-panel kontur sebagai *inetrface* dengan pengguna.

   Sistem aplikasi akan memanggil modul untuk memvisualisasikan layar utama yang dinamakan *'Contour Map Chart*' sebagai *interface* dengan pengguna

4.1.2 Analisa Kebutuhan Data

Menggambarkan kebutuhan data yang diperlukan dalam beroperasi dan harus berfungsi dengan baik.

1. Data Lokasi masing-masing sumur minyak pada suatu lapangan minyak.

   Berisi data Lokasi X dan Y suatu sumur minyak. Data Lokasi x dan y merupakan data sampel yang digunakan dalam proses kriging.

2. Data besar produksi minyak ditiap sumur.

   Berisi data produksi minyak sumur minyak. Data produksi sumur Lokasi x dan y merupakan data sampel yang digunakan dalam proses kriging.

4.2 Perancangan Sistem Aplikasi

Aplikasi peta *Countur* yang diberi nama "Oil System v1.1" yang disesuaikan dengan sistem sebelumnya yang bernama Oil System v1.0 adalah aplikasi yang mengambil data produksi minyak untuk dianalisa dengan persamaan matematika algoritma kriging yang membentuk titik-titik kontur yang ditampilkan menjadi sebuah peta kontur *2D.*

Layanan-layanan yang dapat digunakan dalam aplikasi ini seperti :

1. Data input, untuk membaca data dari database atau file teks

2. Grafik Chart, menampilkan perkembangan suatu sumur minyak dalam beberapa tahun yang digambarkan dalam sebuah grafik yang berbentuk *Line chart, bar chart, spider chart,* dan *pie chart.*
3. Pemetaan kontur, untuk melakukan proses kriging dan menampilkannya dalam bentuk *image countur 2D* dan menyimpan image tersebut menjadi file gambar yang berekstensi *.png.

4.2.1 Perancangan Komponen

Perancangan komponen merupakan tahap dalam menganalisa suatu sistem yang akan dibuat untuk mendapatkan rancangan sistem yang dapat dirancang ulang kembali dan lebih baik dengan cara visualisai, kontruksi dan mendokumentasi hal-hal yang terdapat dalam suatu sistem yang akan dibangun (*Suhendar, 2002*) . Metode yang digunakan adalah pendekatan berorientasi objek (*object oriented*) dengan menggunkan konsep UML (*unified modelling language*) dengan menggunakan *system tool Rational Rose.*

Dalam pembuatan perangkat lunak dibutuhkan visual model dari perangkat lunak yang akan dibuat. Dalam bagian ini dilakukan analisis dan perancangan menggunakan diagram objek yang menggambarkan visual modeling aplikasi Oil System yang terdiri dari beberapa model diantaranya :

1. *Use-case Model,* untuk pemodelan kebutuhan user.
2. *Analysis Model,* untuk pemodelan kebutuhan sistem.
3. *Design Model,* untuk pemodelan kebutuhan dalam pembuatan (pemrograman) aplikasi.

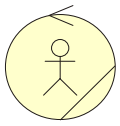
Ada beberapa diagram UML yang digunakan untuk visual modeling pada tugas akhir ini diantaranya :

1. *Use case diagram.*
2. *Class diagram.*
3. *Sequence diagram.*
4. *Collaboration diagram.*
5. *Activity diagram.*

4.2.1.1 Use case Model

Hal pertama yang dilakukan dalam merancang aplikasi Oil System yaitu mengidentifikasi fungsionalitas yang penting secara baik dan terstuktur agar diketahui kebutuhan apa saja yang terdapat dalam sistem. Untuk identifikasi tersebut dapat digambarkan menggunaka *use-case* dan *activity diagram*.

Aplikasi ini digunakan oleh analis perminyakan yang digambarkan dalam notasi UML sebagai *businees worker* yaitu orang yang menjalankan aplikasi ini.

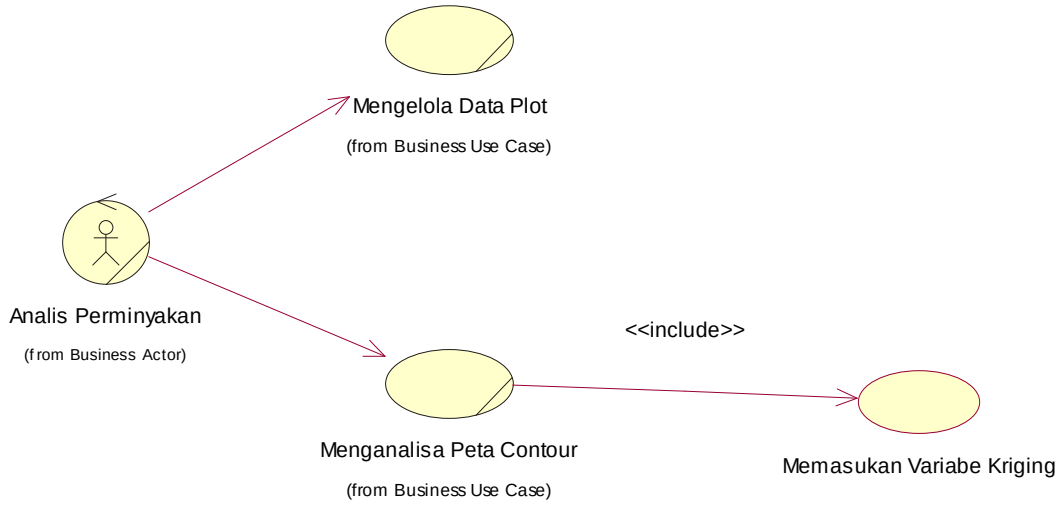


Analis Perminyakan

Gambar 4. 6 *Business Worker* dalam *Use case Model*

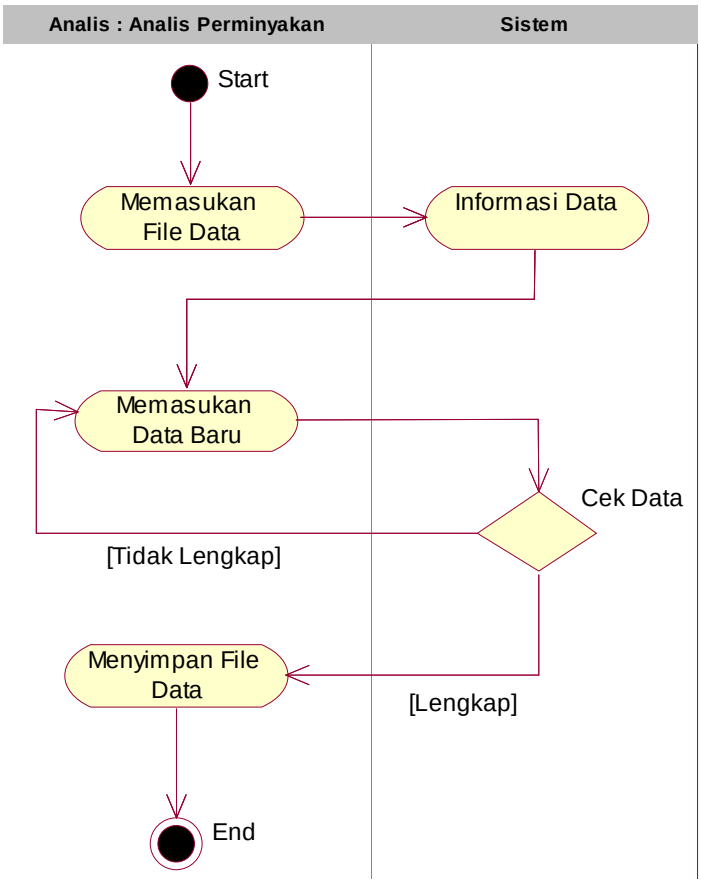
Tugas analis perminyakan (*use case*) adalah memasukan data plot yang akan dianalisa dan data plot tersebut disimpan serta digunakan untuk pembentukan peta kontur dengan melakukan perhitungan kriging dimana telah

dilakukan penginputan variabel-variabel kriging sebelumnya. Hubungan tugas-tugas tersebut digambarkan dengan diagram dibawah ini.



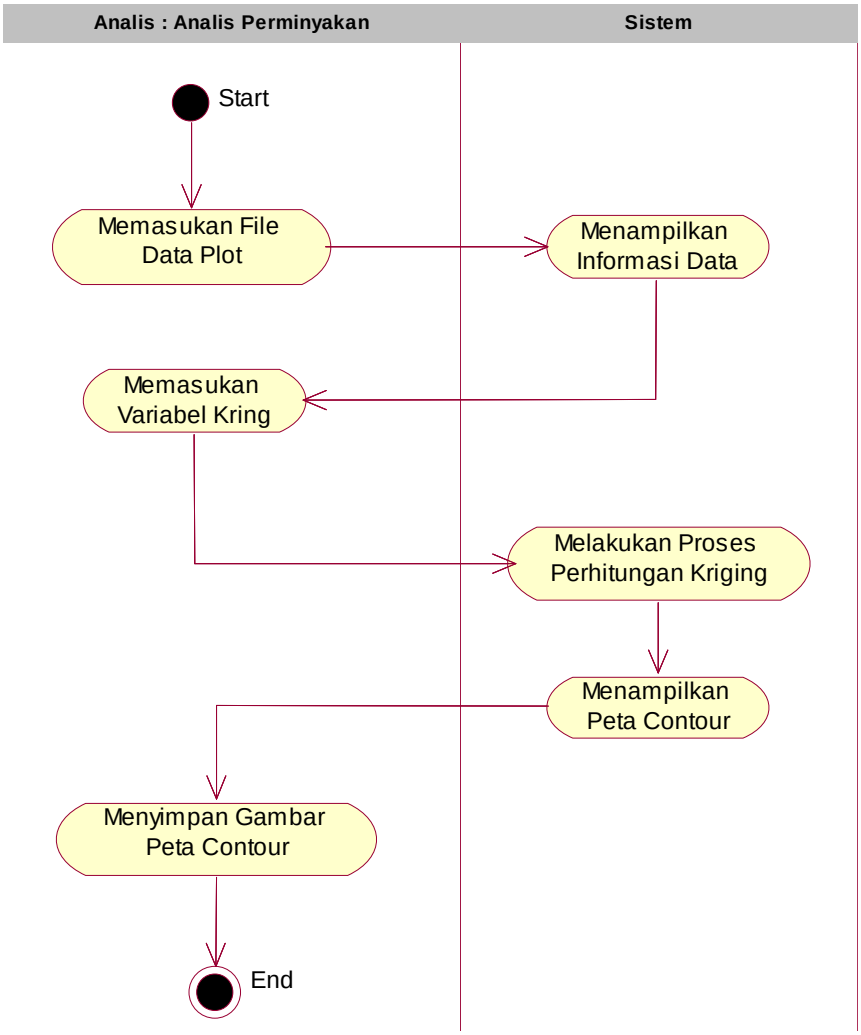
Gambar 4. 7 Aliran use case diagram

Untuk menjelaskan secara detil alur kerja (*workflow*) suatu *use case*, dapat kita gunakan *activity diagram.* Diagram ini memodelkan sebuah alur kerja dari suatu aktifitas ke aktifitas lainnya dari aktifitas keadaan sesaat (*state*). Dengan menggunakan *activity diagram* dapat dijelaskan bagaimana proses dalam *use case* Mengelola Data Plot dan Menganalisa Peta Kontur berinteraksi.



Gambar 4. 8 *Activity Diagram* Mengelola Data Plot

Dari diagram diatas dapat dijelaskan aktifitas yang dilakukan oleh analis perminyakan adalah memasukan file data, memasukan data plot baru, dan menyimpan file data tersebut dalam bentuk file teks yang memudahkan dalam merubah isi file data dengan aplikasi notepad. Sedangkan aktifitas yang dilakukan sistem diantaranya menampilan informasil isi file data dan mencek apakah semua data yang dibutuhkan telah lengkap atau belum karena apabila tidak lengkap akan diperintahkan kepada analis untuk memasukan data dengan lengkap karena akan mempengaruhi pembacaan file teks dikemudian hari.



Gambar 4. 9 *Activity Diagram* Menganalisa Peta Kontur

Dari diagram diatas dapat dijelaskan aktifitas yang dilakukan oleh analis perminyakan adalah memasukan file data, memasukan perubahan variabel-variabel kriging yang sesuai dengan yang diinginkan, dan menyimpan peta kontur tersebut menjadi file image agar dapat dilihat kembali untuk dianalisa lang. Sedangkan aktifitas yang dilakukan sistem diantaranya menampilan informasil isi file data, melakukan perhitungan kriging dan menampilkan hasil perhitungan kriging menjadi peta kontur sumur minyak.

4.2.1.2 Analisis Model

Model Analisis (*Analysis Model)* menggambarkan realisasi dari model *use case* yang telah dirancang sebelumnya, sebagai abstraksi pemetaan awal perilaku dalam sistem aplikasi Oil System kedalam elemen-elemn pemodelan. Kelas-kalas dalam analisis ini akan dikembangkan menjadi elemen-elemen model desain. *Visual Modelling* yang digunakan dalam model ini adalah *sequence diagram, collaboration diagram,* dan *class diagram.*

4.2.1.2.1 Mengelola Data Plot

Dalam *use case* mengelola data plot terdapat interaksi antar objek yang disusun berdasarkan urutan waktu yang digambarkan melalui *sequence diagram* agar dapat melihat tahapan yang terjadi untuk menghasilkan *use case* yang benar.

Kelas-kelas analisi dapat digambarkan menjadi beberapa kelas sebagai berikut :

1. Kelas *Boundary*

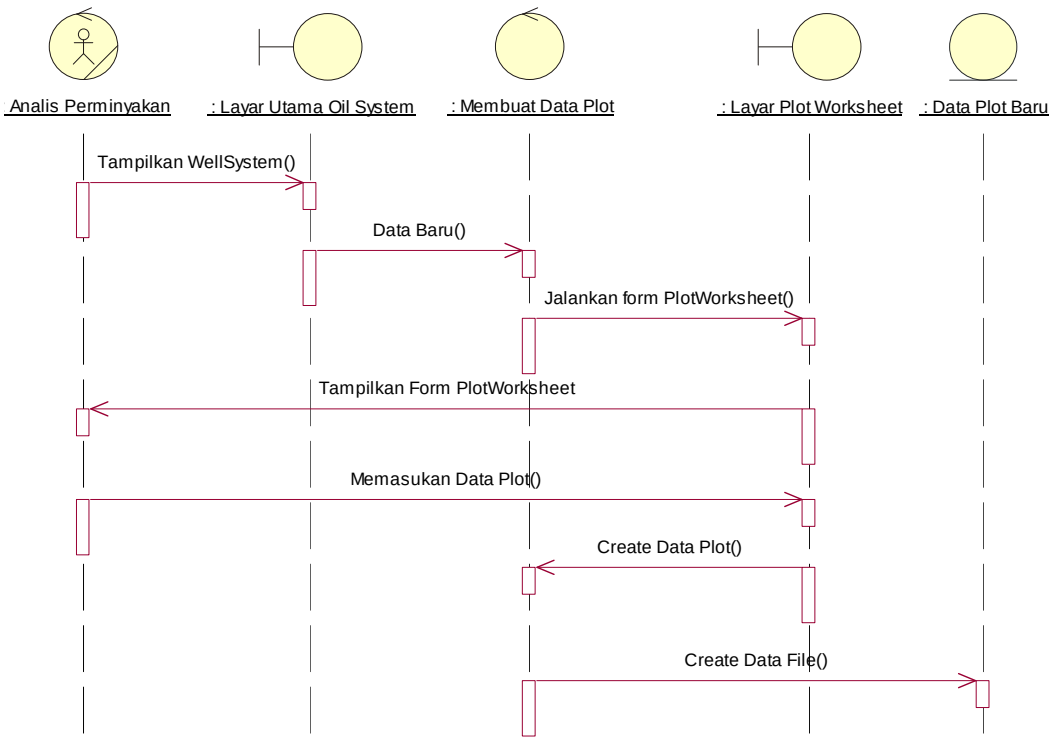
   Kelas yang menggambarkan interaksi antara *actor* analisis perminyakan dengan sistem seperti *User Interface* Layar Utama Oil System dan Layar Plot worksheet.

2. Kelas *Control*

   Kelas yang mengontrol alur kerja sistem seperti Membuat Data Plot.

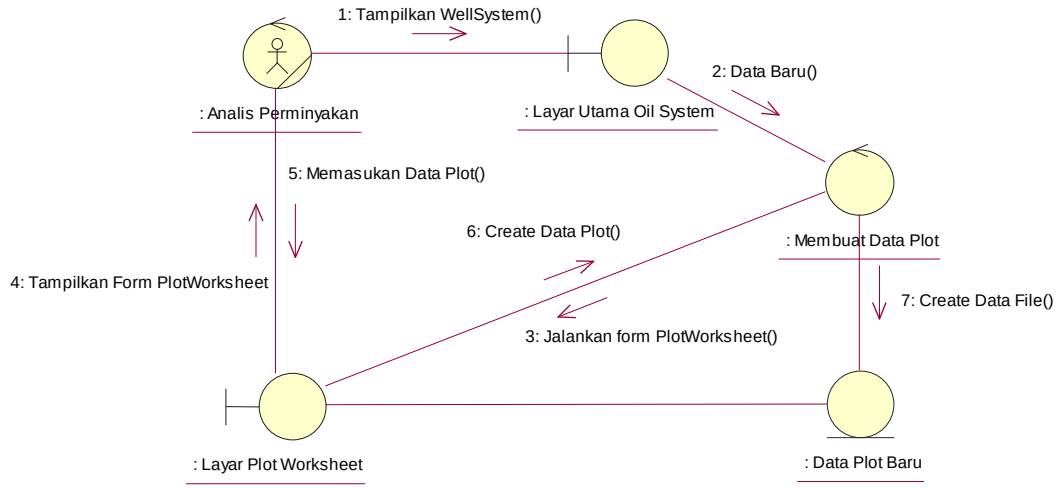
3. Kelas *Entity*

   Kelas yang menggambarkan informasi yang harus disimpan dalam sistem seperti file Data Plot baru.



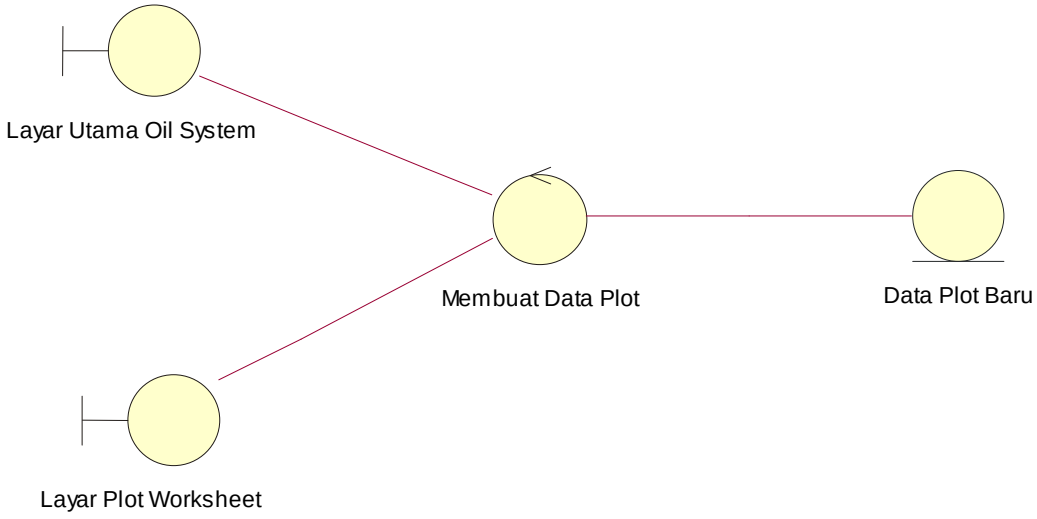
Gambar 4. 10 *Sequence Diagram* untuk menjelaskan realisasi *use case* mengelola data plot.

Dari *Sequence diagram* diatas, kita dapat mengembangkan menjadi *Collaboration diagram* untuk menggambarkan interaksi yang mengungkapkan keputusan mengenai perilaku sistem. Hubungan yang terjadi antar kelas-kelas dapat dilihat pada diagram dibawah ini.



Gambar 4. 11 *Collaboration Diagram* untuk menjelaskan Realisasi *Use case* Mengelola data plot.

Kelas-kelas analisis yang telah dibuat dan digunakan pada kedua diagram diatas, dapat digunakan untuk membuat *Class Diagram* untuk menggambarkan hubungan antar kelas untuk model desain.



Gambar 4. 12 *Class Diagram* untuk menjelaskan realisasi *Use case* Mengelola data plot.

4.2.1.2.2 Menganalisa Peta *Contour*

Dalam *use case* menganalisa peta Kontur terdapat interaksi antar objek yang disusun berdasarkan urutan waktu yang digambarkan melalui *sequence diagram* agar dapat melihat tahapan yang terjadi untuk menghasilkan *use case* yang benar.

Kelas-kelas analisis dapat digambarkan menjadi beberapa kelas sebagai berikut :

1. Kelas *Boundary*

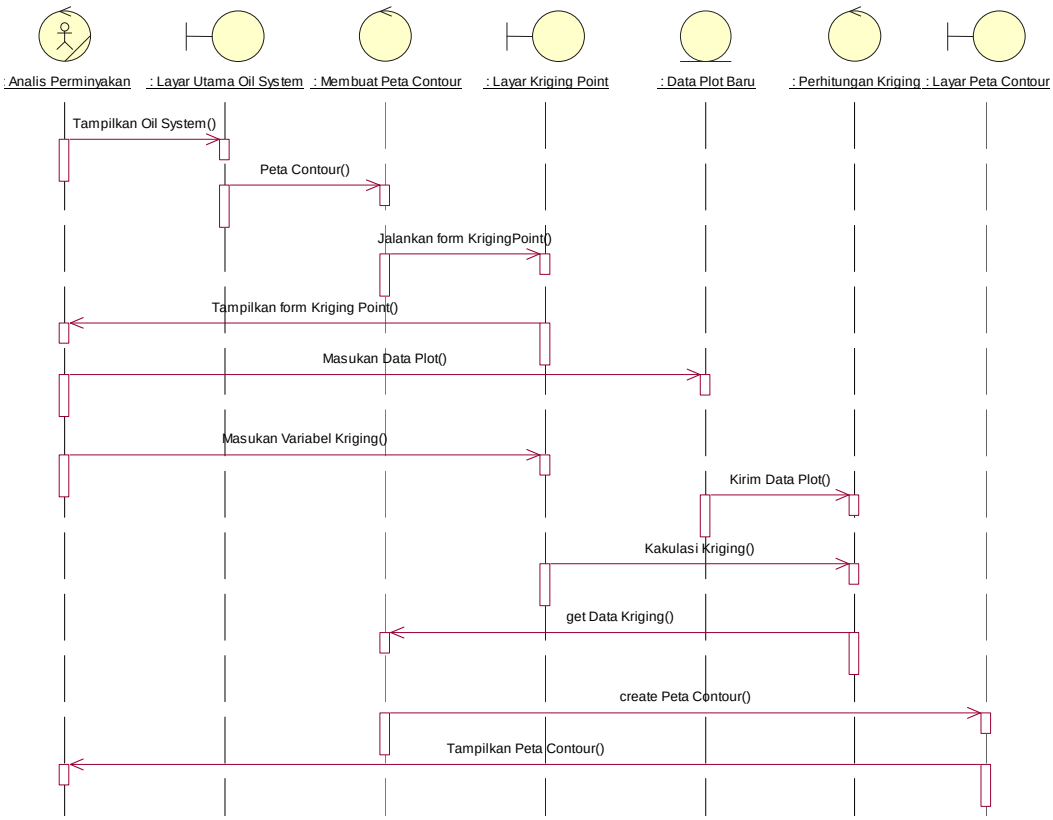
   Kelas yang menggambarkan interaksi antara *actor* analisis perminyakan dengan sistem seperti *User Interface* Layar Utama Oil System, Layar *Kriging Point*, dan Layar Peta Kontur.

2. Kelas *Control*

   Kelas yang mengontrol alur kerja sistem seperti Membuat Peta Kontur dan Perhitungan kriging.

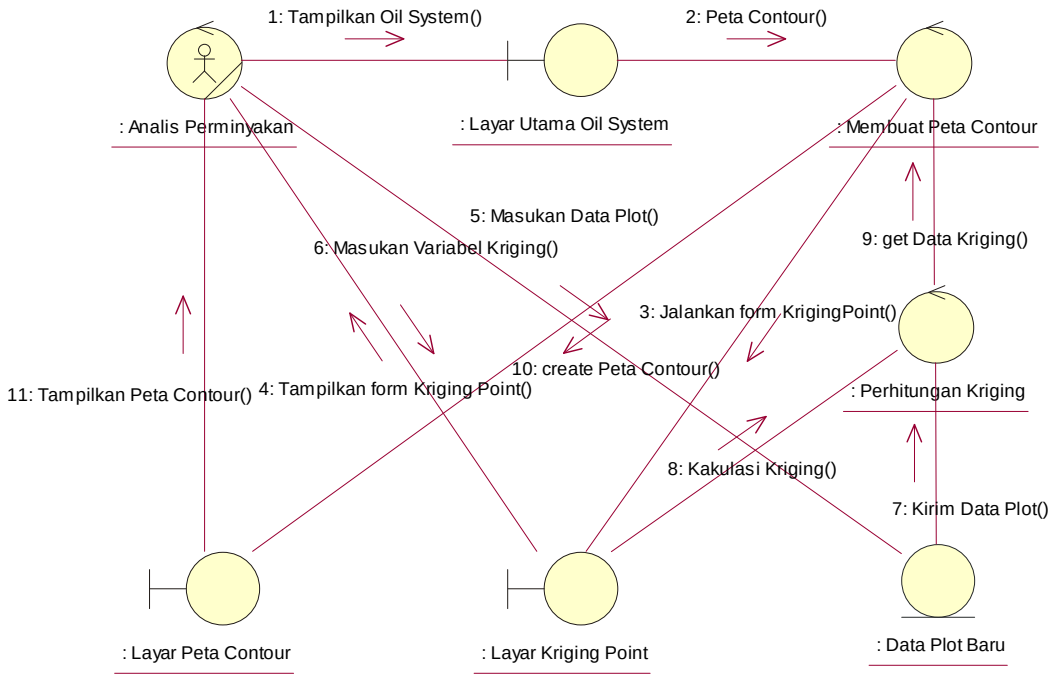
3. Kelas *Entity*

   Kelas yang menggambarkan informasi yang harus disimpan dalam sistem seperti file Data Plot baru.



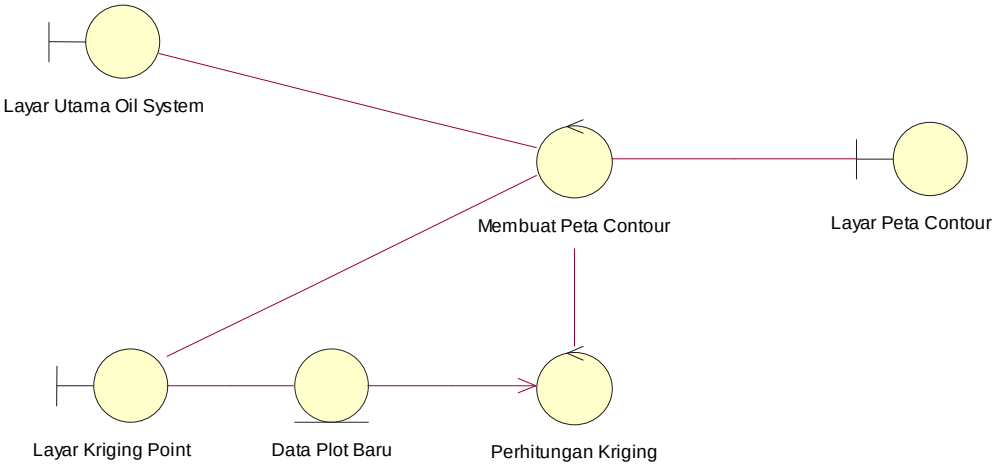
Gambar 4. 13 *Sequence Diagram* untuk menjelaskan realisasi *use case* Menganalisa peta kontur.

Dari *Sequence diagram* diatas, kita dapat mengembangkan menjadi *Collaboration diagram* untuk menggambarkan interaksi yang mengungkapkan keputusan mengenai perilaku sistem. Hubungan yang terjadi antar kelas-kelas dapat dilihat pada diagram dibawah ini.



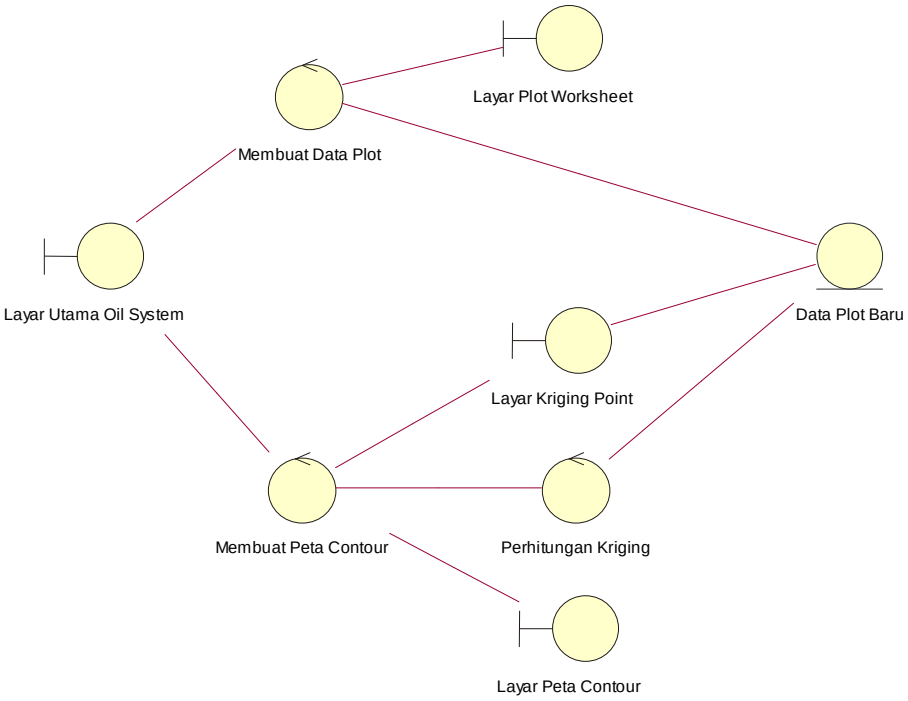
Gambar 4. 14 *Collaboration Diagram* untuk menjelaskan Realisasi *Use case* Menganalisa peta kontur.

Kelas-kelas analisis yang telah dibuat dan digunakan pada kedua diagram diatas, dapat digunakan untuk membuat *Class Diagram* untuk menggambarkan hubungan antar kelas untuk model desain.



Gambar 4. 15 *Class Diagram* untuk menjelaskan realisasi *Use case* Menganalisa peta kontur.

Tahap akhir dalam analisi model ini adalah menggabungkan semua kelas yang ada di *use case* mengelola data plot dan *use case* membuat peta kontur yang digambarkan dalam sebuah *Classs Diagrami* berikut :

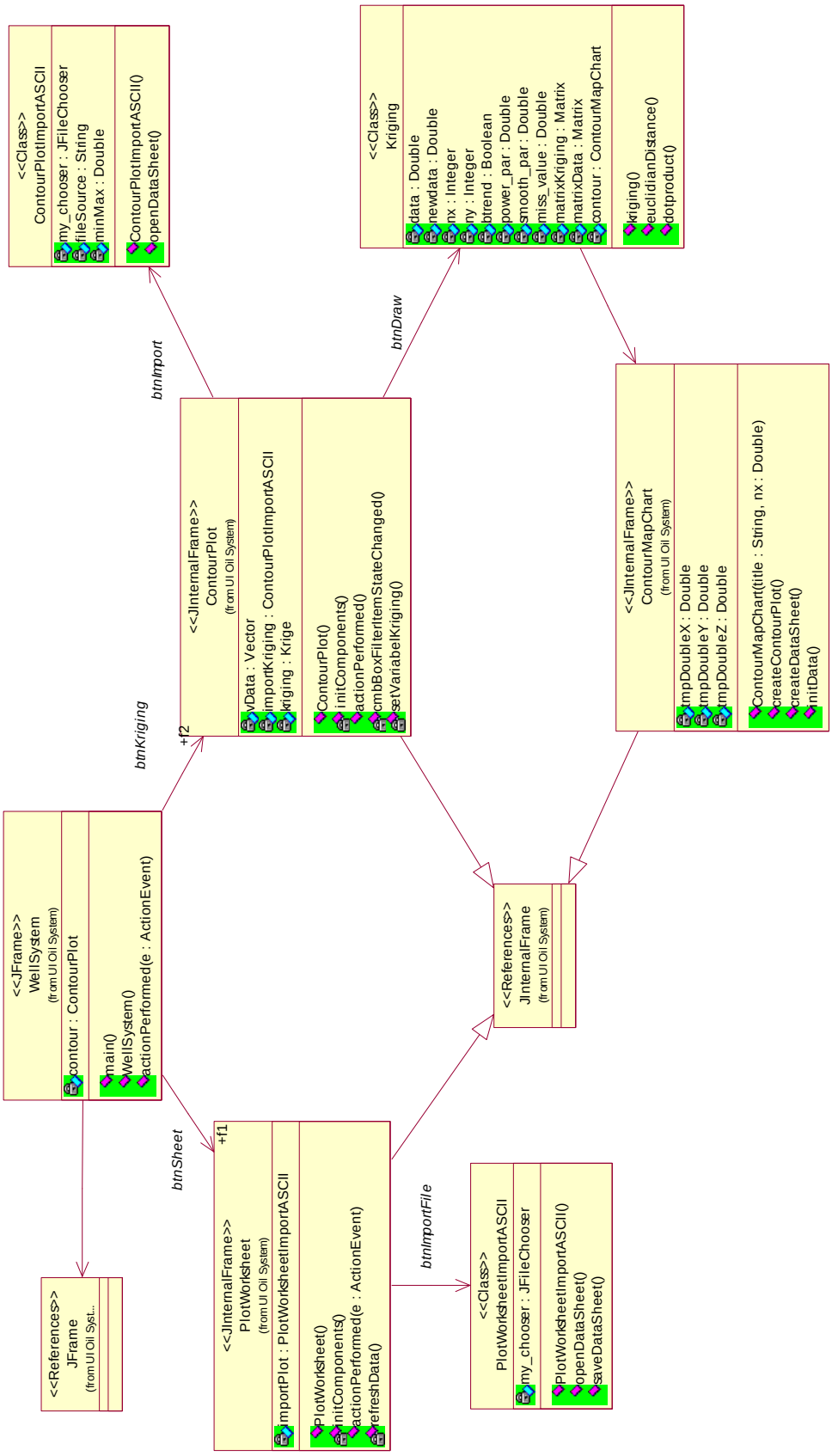


Gambar 4. 16 *Class Diagram* untk menjelaskan Realisasi *Use case-Use case* dalam Aplikasi Oil System.

Kelas-kelas dalam model analisis memiliki urutan keadaan sesaat (*state*) yang digambarkan menggunakan *Statechart Diagram* pada lampiran A.

4.2.1.3 Design Model

Model Desain (*Design Model)* menggambarkan abstraksi perilaku aplikasi dalam penerapan (implementasi) dari aplikasi Oil System yang lebih mendekati *source code* berdasarkan kelas-kelas dalam model analisis yang telah dibuat. *Visual Modelling* yang digunakan dalam model ini adalah *Class Diagram* dan *Component Diagram.*

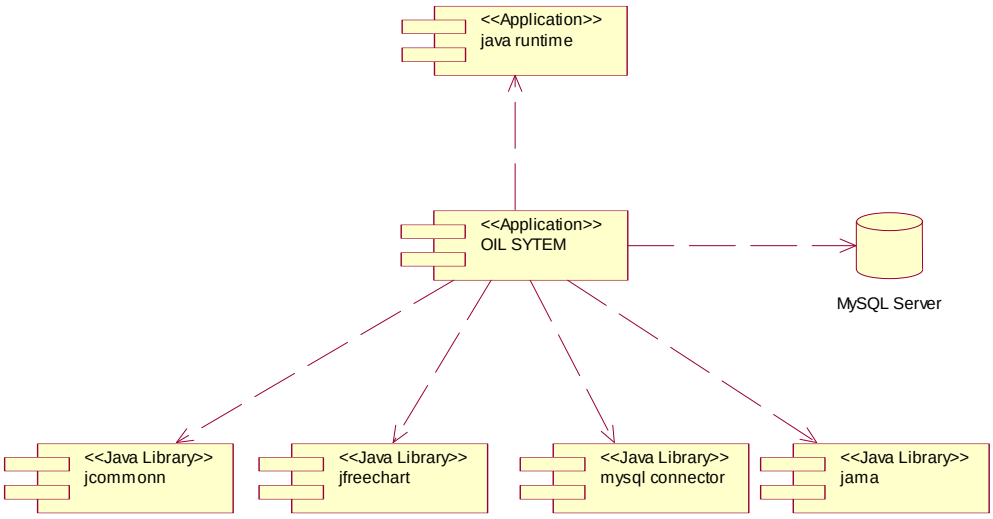

<<Class>>
ContourPlotImportASCII
my_chooser : JFileChooser
fileSource : String
minMax : Double
ContourPlotImportASCII()
openDataSheet()
<<Class>>
Kriging
data : Double
newdata : Double
nx : Integer
ny : Integer
otrend : Boolean
power_par : Double
smooth_par : Double
miss_value : Double
matrixKriging : Matrix
matrixData : Matrix
contour : ContourMapChart
kriging()
euclidianDistance()
dotproduct()
btnImport
btnDraw
<<JInternalFrame>>
ContourPlot
(from UI Oil System)
vData : Vector
importKriging : ContourPlotImportASCII
kriging : Krige
ContourPlot()
initComponents()
actionPerformed()
cmbBoxFilterItemStateChanged()
setVariabelKriging()
<<JInternalFrame>>
ContourMapChart
(from UI Oil System)
tmpDoubleX : Double
tmpDoubleY : Double
tmpDoubleZ : Double
ContourMapChart(title : String, nx : Double)
createContourPlot()
createDataSheet()
initData()
+f2
btnKriging
<<JFrame>>
WellSystem
(from UI Oil System)
contour : ContourPlot
main()
WellSystem()
actionPerformed(e : ActionEvent)
<<References>>
JInternalFrame
(from UI Oil System)
btnSheet
+f1
<<JInternalFrame>>
PlotWorksheet
(from UI Oil System)
importPlot : PlotWorksheetImportASCII
PlotWorksheet()
initComponents()
actionPerformed(e : ActionEvent)
refreshData()
btnImportFile
<<Class>>
PlotWorksheetImportASCII
my_chooser : JFileChooser
PlotWorksheetImportASCII()
openDataSheet()
saveDataSheet()
<<References>>
JFrame
(from UI Oil Syst...

Kelas-kelas dalam *Class Diagram* terbagi atas :

1. Kelas *Baundary*, Kelas yang berhubungan dengan *user interface* dalam sistem, seperti Layar Utama Well System (WellSystem), Layar PlotWorksheet, Layar *Kriging Point* (ContourPlot), dan Layar Peta Kontur (ContourMapChart).
2. Kelas *Entity*, Kelas yang berhubungan dengan informasi yang diolah dalam sistem, seperti PlotWorksheetASCII, ContourPlotASCII, dan Kriging.

Component Diagram merupakan diagram yang mengambarkan hubungan komponen yang dibutuhkan dalam sistem diantaranya :

1. Java *runtime* untuk menjalankan aplikasi.
2. Jcommon.jar dan jfreechart.jar untuk proses pembentukan peta kontur.
3. Mysql server untuk penyimpanan data dan mysql connector.jar untuk koneksi data ke sistem.
4. Jama.jar untuk membentuk matrik dalam perhitungan kriging.



Gambar 4. 17 Diagram komponen

4.2.2 Perancangan Basisdata

Fungsi kamus data yaitu membuat detail data yang akan dipersiapkan pada tahap implementasi selanjutnya.

1. Tabel Sumur

   Nama : WellHeader

   Deskripsi isi : Berisi data sumur

   Primary key : WellID

   Foreign Key : WellID

Tabel 4. 5 Data WellHeader

Nama Field	Type dan Length	Deskripsi	Boleh Null	Default
*WellID**	Varchar, 10	ID Sumur	No	AutoInc
WellName	VarChar,25	Nama Sumur	No	-
Loc_X	Double	Lokasi X di suatu lapangan minyak	No	-
Loc-Y	Double	Lokasi X di suatu lapangan minyak	No	-
StartDepth	Double	Awal Perhitungan Kedalaman Sumur	Yes	-
StopDepth	Double	Akhir Perhitungan Kedalaman Sumur	Yes	-
KB	Int	Data sumur	Yes	-
WellStatus	Varchar, 20	Status sumur	Yes	-
CreateDate	Date	Waktu eksplorasi	Yes	-
CreateBy	Varchar, 50	Pembuat Sumur	Yes	-

2. Tabel Produksi Sumur

   Nama : WellProd

   Deskripsi isi : Berisi data produksi sumur harian

   Primary key : prodID

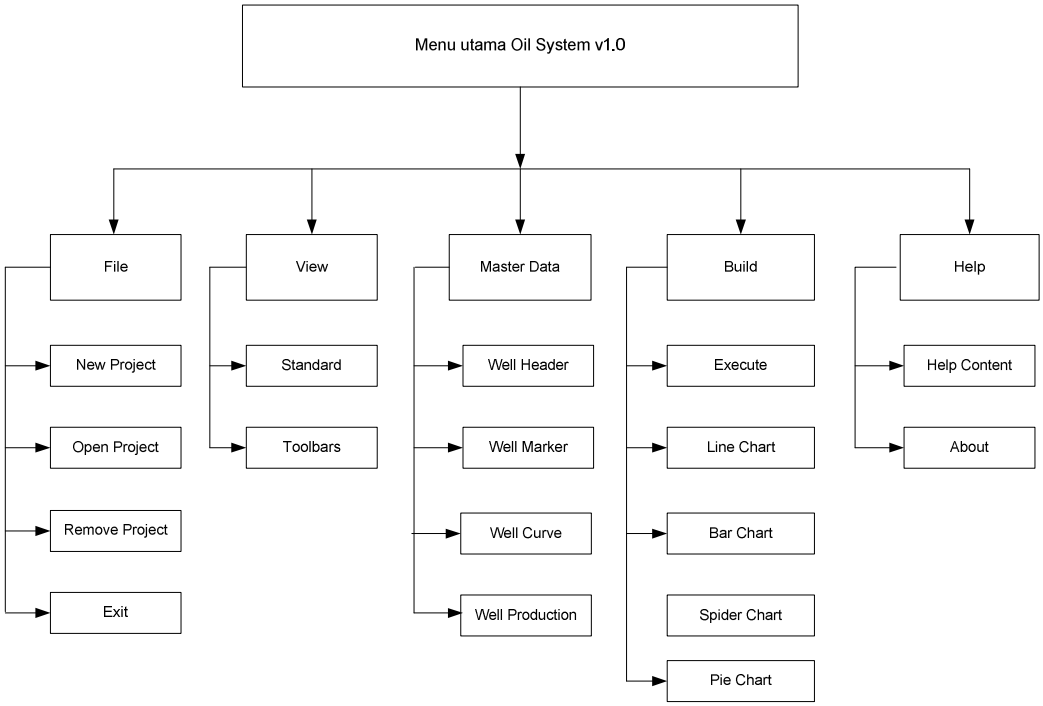
   Foreign Key : WellID

Tabel 4. 6 Data WellProd

Nama Field	**Type dan Length**	**Deskripsi**	**Boleh Null**	**Default**
*prodID**	int	ID Produksi	No	AutoInc
WellID	VarChar,10	ID Sumur	No	-
OilProd	Double	Besar Produksi Minyak	No	
WaterProd	Double	Besar produksi air	No	-
Pressure	Double	Besar Tekanan	No	-
WellHeat		Data Produksi	No	
Ondate	Double	Waktu Produksi	No	

4.2.3 Rancangan Struktur Menu

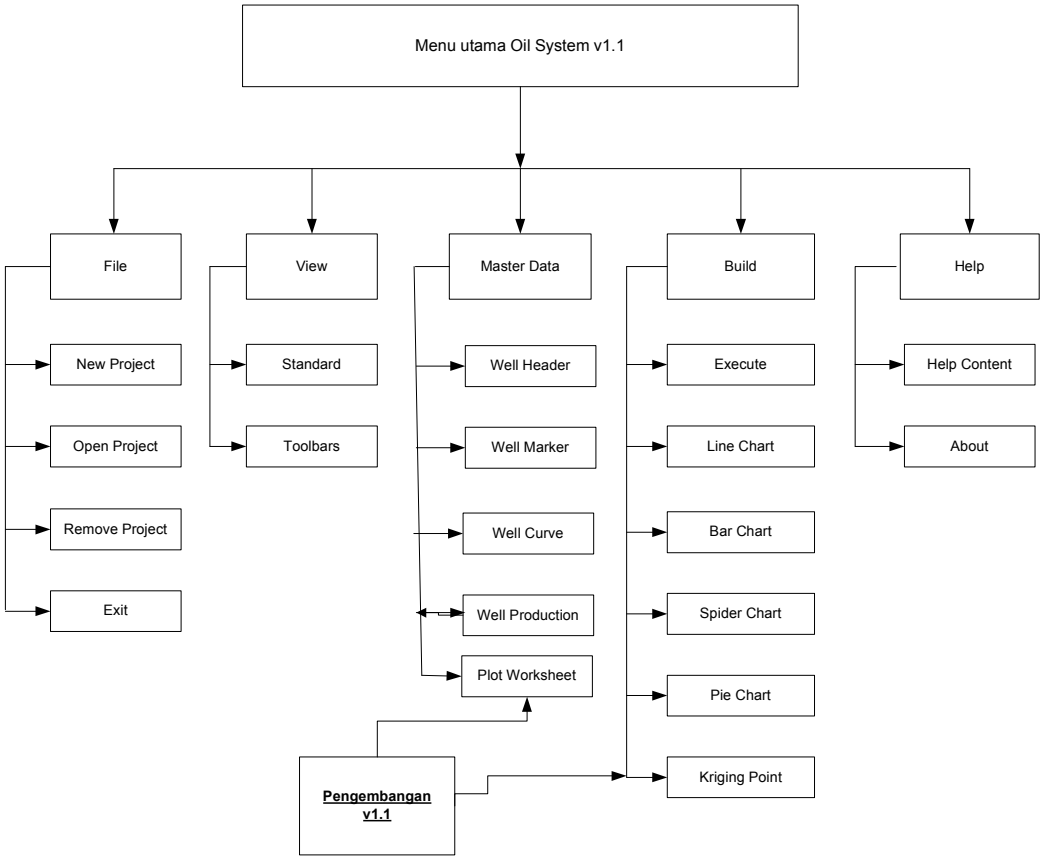
Berikut adalah struktur menu dari Oil System v1.0 sebelumnya :



Gambar 4. 18 Struktur Menu Oil System v1.0

Dari Gambar 4.13, dapat dilihat pada menu "*Build*" hanya terdapat 5 sub menu. Penulis akan menambahkan dua sub menu pada menu build yaitu "*Calculate Semivariogram*" untuk melakukan proses perhitungan semivariogram dan "*Kriging Point*" untuk melakukan proses perhitungan kriging.

Berikut adalah perancangan struktur menu dari Oil System v1.0 yang penulis tambahkan dalam menu Oil System v1.1:



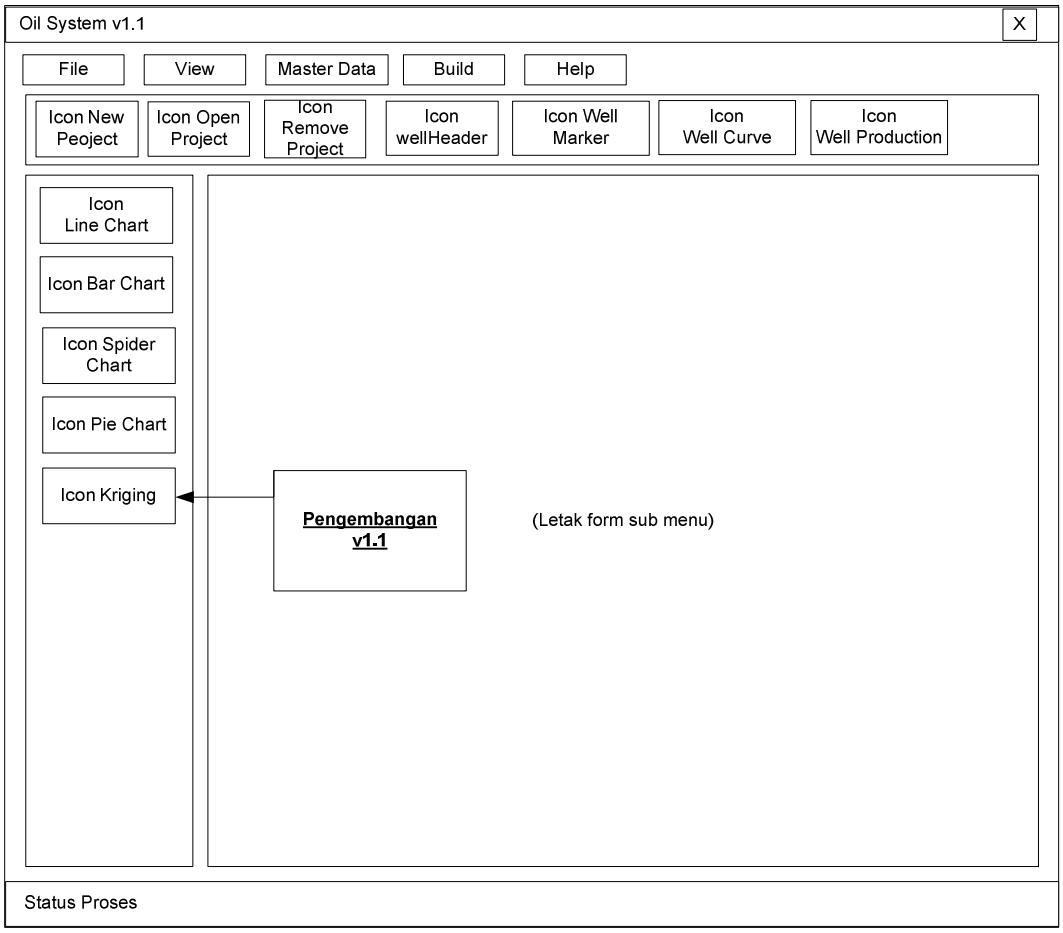
Gambar 4. 19 Struktur Menu Oil System v1.1

4.2.4 Perancangan Tampilan

Tampilan yang akan dirangcang terdiri dari rancangan untuk tampilan menu utama dan *kriging point*.

4.2.4.1 Modul Menu Utama

Bentuk rancangan menu utama antar muka yang akan ditampilkan adalah:



Gambar 4. 20 Rancangan Tampilan Oil System

Modul Menu Utama ini merupakan modul yang berfungsi untuk melakukan integrasi antar modul, dalam modul utama ini terdapat 5 menu yaitu menu file yang berfungsi untuk membuat project analisa, kemudian menu view untuk mengganti tampilan *toolbar*, menu master data berfungsi untuk melakukan

input data sumur. Selanjutnya adalah menu build berfungsi melakukan analisa grafik chart dan kontur 2D, yang terakhir adalah menu *help* yang menerangkan tentang informasi penggunaan dan pengembang sistem.

4.2.4.2 Modul Menu Master Data

Modul Menu Data Master ini dikelompokan menjadi 5 sub menu yaitu proses pengelolaan data *Well Header*, data *Well Curve*, data *Well Marker*, dan data *Well Production*, dan *Plot Worksheet.*

4.2.4.2.1 Modul *Plot Worksheet*

Gambar 4. 21 Rancangan Tampilan form Plot Worksheet

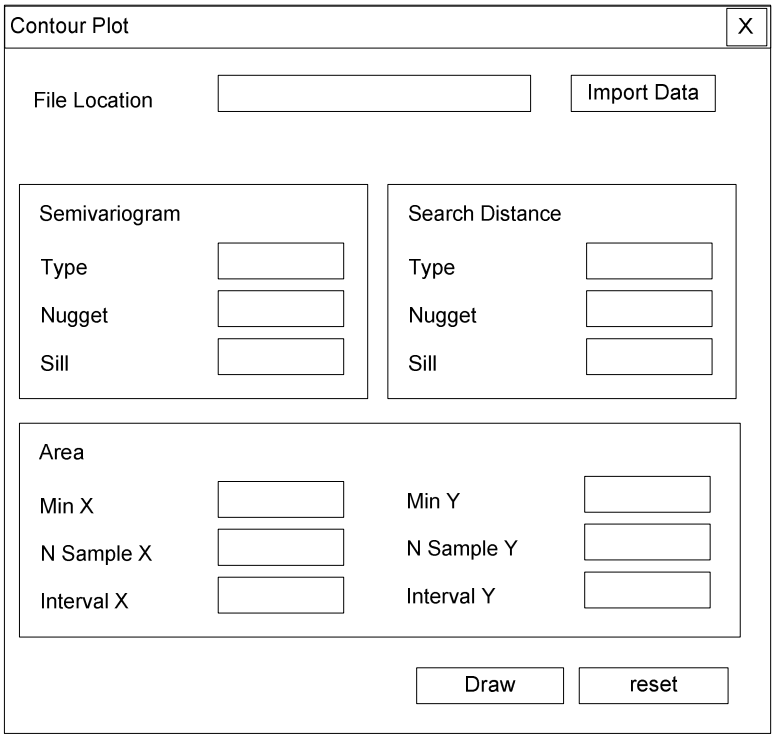
Modul Menu *Plot Worksheet* ini merupakan modul yang berfungsi untuk melakukan input nilai-nilai data lokasi x, lokasi y, dan variable yang digunakan

seperti *Oil Production*, *Water Production*, *Pressure*, dan *Well Heat*. Data dapat diimport dan disimpan dalam bentuk data teks.

4.2.4.3 Modul Menu Build

Modul Build dikelompokan menjadi tiga yaitu execute, *graphic chart* yang terdiri dari *Line Chart, Bar Chart, Spider Chart*, dan *Pie Chart*, kemudian yang terakhir *kriging point*.

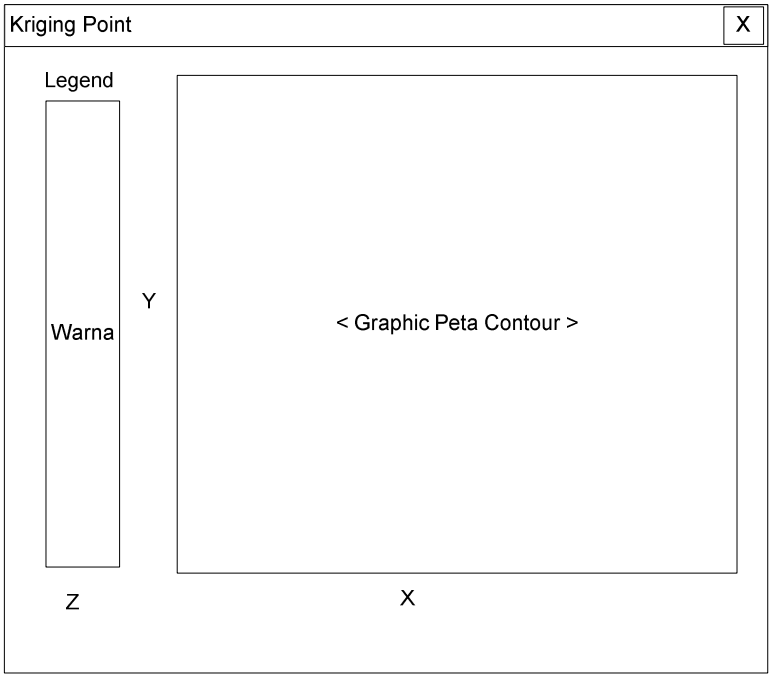
4.2.4.3.1 Modul *Contour Plot*



Gambar 4. 22 Rancangan Tampilan form *Contour Plot*

Modul *Contour Plot* merupakan modul yang berfungsi untuk melakukan import data-data nilai x,y, dan z serta input variabel-variabel yang mempengaruhi terbentuknya peta kontur yang diingingkan.

4.2.4.4 Modul Menu *Kriging Point*



Gambar 4. 23 Modul *Kriging Point*

Modul Menu *Kriging Point* merupakan modul yang berfungsi untuk menampilkan grafik kontur sumur minyak yang disertai dengan parameter legend untuk mengetahui tingkat grafik kontur dengan variabel yang digunakan.

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

Implementasi merupakan tahap dimana sistem siap dioperasikan pada keadaaan yang sebenarnya, sehingga akan diketahui sistem yang dibuat benar-benar dapat menghasilkan tujuan yang ingin dicapai.

5.1.1 Batasan Implementasi

Batasan implementasi dari Tugas Akhir ini adalah sebagai berikut:

1. Metode Kriging yang digunakan hanya *Punctual* (*Ordinary*) *Kriging.*
2. Peta kontur hanya berbentuk 2D (2 Dimensi)
3. Data produksi yang digunakan dalam perhitungan algoritma kriging hanya berupa data produksi minyak.

5.1.2 Lingkungan implementasi

Pada prinsipnya setiap desain sistem yang telah dirancang memerlukan sarana pendukung yaitu berupa peralatan-peralatan yang sangat berperan dalam menunjang penerapan sistem yang didesain terhadap pengolahan data. Komponen-komponen yang dibutuhkan antara lain *hardware*, yaitu kebutuhan perangkat keras komputer dalam pengolahan data kemudian *software*, yaitu kebutuhan akan perangkat lunak berupa sistem untuk mengoperasikan sistem yang telah didesain.

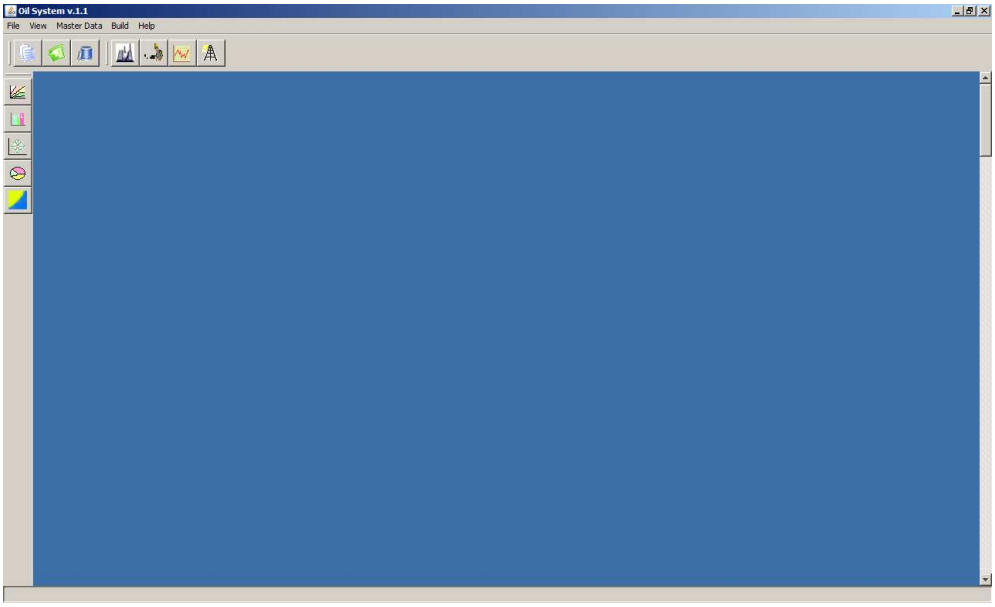
1. Perangkat Keras
   a. *Processor* : Intel Core 2 Duo E6550 2.33 GHz
   b. *Memory* : 2 GB
   c. *Harddisk* : 200 GB
2. Perangkat Lunak
   a. Sistem Operasi : Windows XP Profesional SP2
   b. Bahasa Pemrograman : *Java*
   c. Tools : Eclipse 3.0, Netbean 6.0 dan Xampp 1.6.1
   d. DBMS : *MySQL*

5.1.3 Hasil Implementasi

Pada implementasi aplikasi Oil System, terdapat beberapa modul diantaranya modul utama, modul data sheet, modul *Contour plot*, dan modul *kriging point*.

5.1.3.1 Modul Menu Utama

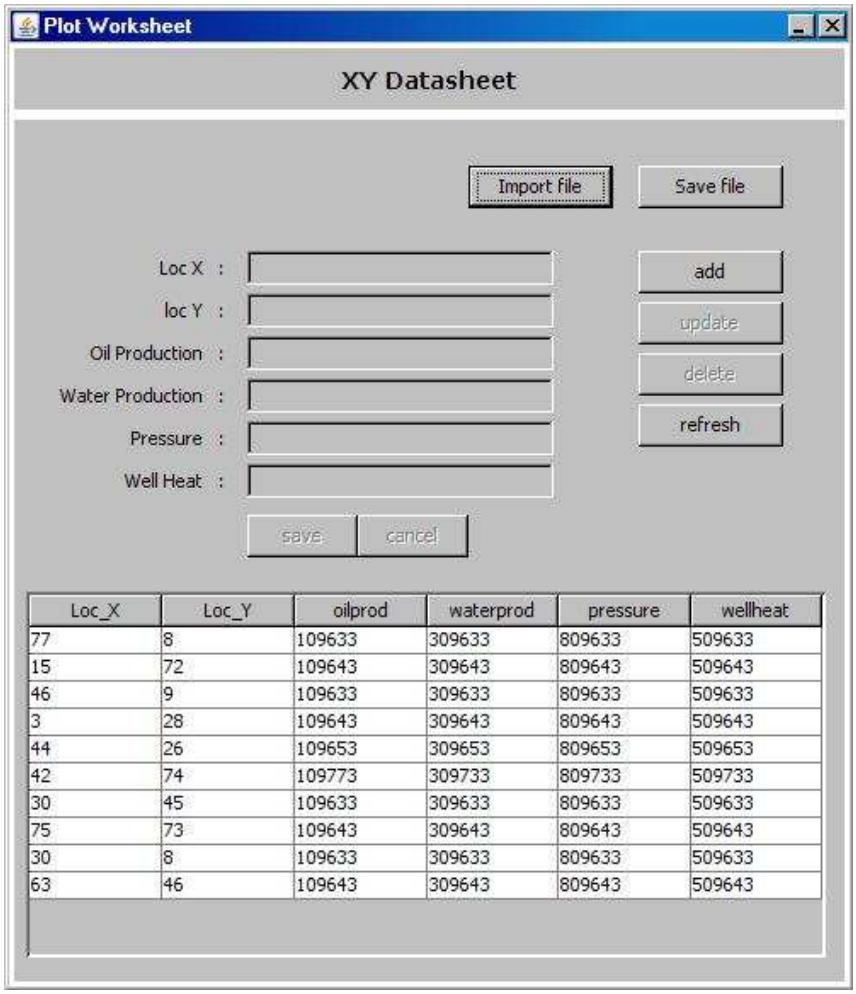
Modul Utama dari Aplikasi Oil System merupakan modul yang berfungsi untuk melakukan integrasi antar modul. Modul utama dapat berupa tampilan berikut ini:



Gambar 5. 1 Tampilan Modul New Menu Utama

Modul utama ini memiliki 5 menu yaitu menu File yang berfungsi untuk pengelolaan project, menu view yang berfungsi untuk model tampilan form kemudian menu master data yang berfungsi untuk input data sumur. Selanjutnya menu build untuk melakukan analisa terhadap grafik chart dan peta kontur, yang terakhir adalah menu help yang menerangkan tentang cara menggunakan program.

5.1.3.2 Modul *Plot Worksheet*



Loc_X	Loc_Y	oilprod	waterprod	pressure	wellheat
77	8	109633	309633	809633	509633
15	72	109643	309643	809643	509643
46	9	109633	309633	809633	509633
3	28	109643	309643	809643	509643
44	26	109653	309653	809653	509653
42	74	109773	309733	809733	509733
30	45	109633	309633	809633	509633
75	73	109643	309643	809643	509643
30	8	109633	309633	809633	509633
63	46	109643	309643	809643	509643

Gambar 5. 2 Tampilan Modul Plot Worksheet

Modul Plot Worksheet ini merupakan modul yang berfungsi untuk melakukan input nilai x, y, dan z yaitu : *Oil Production, water Production, Pressure,* dan *Well Heat.*

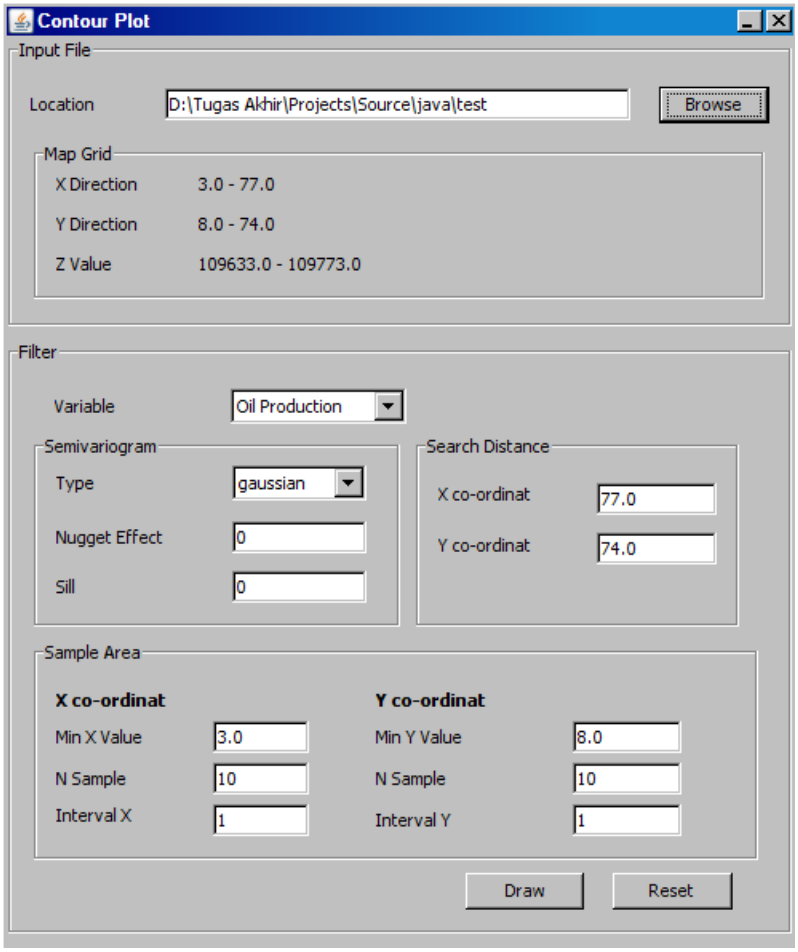
Hal-hal yang dapat dilakukan dalam modul ini sebagai berikut :

1. *Import File* : melakukan input nilai-nilai x, y, dan z dengan mengambilnya dari file data yang sesuai dengan format file yang telah ditentukan.
2. *Save File* : menyimpan nilai-nilai yang diinput kedalam file teks yang isinya sesuai dengan format berikut yang dipisahkan dengan “tab”:

Nilai_X	Nilai_Y	OILPROD	WATERPROD	PRESSURE	WELLHEAT
10	40	1000	2000	3000	4000
20	50	1100	2100	3200	4200
30	60	100	2200	3400	4700

3. *Add* : menambah data input.
4. *Update* : merubah isi data input jika ditemukan perubahan nilai.
5. *Delete* : menghapus data input.
6. *Refresh* : mengembalikan nilai-nilai teks menjadi kosong.

5.1.3.3 Modul *Contour Plot*



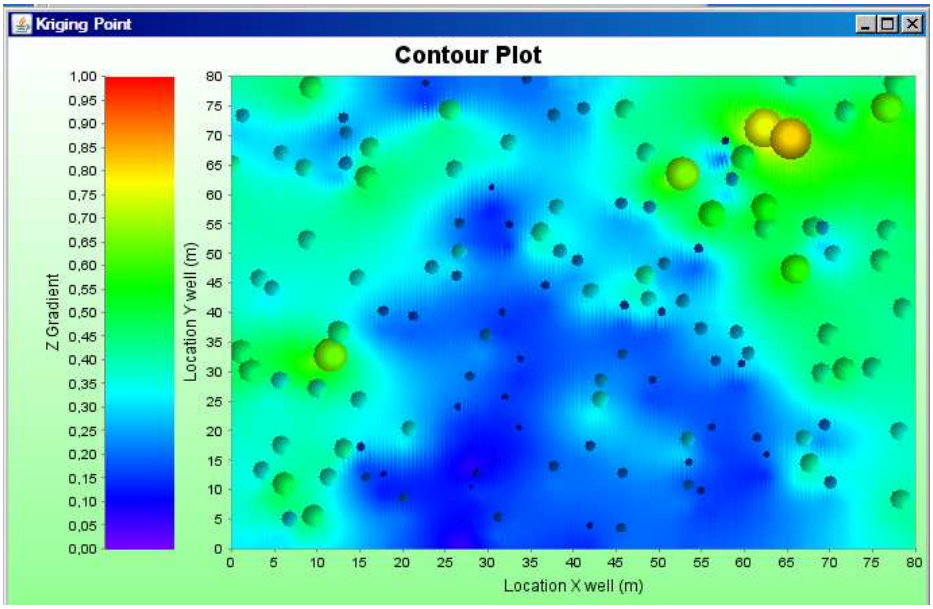
Gambar 5. 3 Tampilan Modul *Contour Plot*

Modul *Contour Plot* berfungsi untuk memasukan nilai x, y dan z dari file teks yang berisi data-data plot, mensetting variabel-variabel yang mempengaruhi dalam perhitungan algoritma kriging yang membentuk peta kontur, dan memanggil Modul *Kriging Point* untuk membentuk peta.

Hal-hal yang dilakukan dalam modul ini sebagai berikut :

1. *Input File* : melakukan input file teks yang berisi data plot dengan menekan tombol browse.
2. Semivariogram : menyeting nilai-nilai membentuk nilai-nilai semivariogram.
   a. *Semivariogram type* : terdiri dari 4 type diantaranya gaussian, spherical, linear, dan exponent.
   b. *Nugget effect* : skala sub grid atau pengukuran error dari variogram
   c. *Sill* : jarak daerah semivariance.
3. *Search Distance* : jarak daerah blok pencarian nilai Z
4. *Sample Area* : Nilai minimal x dan y, dimensi blok x dan y, serta interval x dan y .
5. *Draw* : untuk memanggil fungsi metode kriging dan menampilkannya peta kontur berdasarkan data dan nilai-nilai variabel yang telah dimasukan.
6. *Reset* : mengembalikan nilai awal variabel-variabel kriging berdasarkan input file data.

5.1.3.4 Modul *Kriging Point*



Gambar 5. 4 Modul *Kriging Point*

Modul *Kriging Point* merupakan modul yang berfungsi untuk menampilkan peta kontur dari cadangan minyak pada suatu lapangan minyak.

Adapun cara membaca peta kontur pada modul ini yaitu :

1. *Location X well* (m)

   Angka-angka pada *Location X well* (Lokasi X sumur) menunjukan lokasi x dari cadangan minyak berdasarkan Loc X pada data sampel well header basisdata dalam satuan meter (m).

2. *Location Y well* (m)

   Angka-angka pada *Location Y well* (Lokasi Y sumur) menunjukan lokasi y dari cadangan minyak berdasarkan data Loc X pada data sampel well header basisdata dalam satuan meter (m).

3. Z = *Oil Reservoir* (barel)

   Warna-warna pada Z = Oil Reservoir (cadangan minyak) menunjukan tingkatan cadangan minyak pada suatu lapangan minyak. Dari warna-warna tersebut dapat dijelaskan:

   Tabel 5. 1 Tingkat warna kandungan cadangan minyak

No	Warna	Keterangan
1	Merah	Sangat Tinggi
2	Kuning	Tinggi
3	Hijau	Normal
4	Ungu	Rendah
5	Biru	Sangat Rendah

4. *Buble* sumur

   Menunjukan lokasi sumur-sumur sampel data yang diketahui dan digambarkan dengan warna dan ukuran yang berbeda berdasarkan nilai Z data yang diketahui.

Dari hasil peta kontur tersebut sumur yang terletak pada jarak 60 hingga 70 di *Location X Well* dan jarak 65 hingga 75 di *Location Y Well* memiliki perkiraan kandungan minyak yang sangat tinggi. Hal ini dapat dilihat pada lokasi tersebut ditandai dengan warna merah yang memiliki nilai Z (Oil Resevoir = Cadangan Minyak) tertinggi.

5.2 Pengujian Sistem

Pemrograman merupakan kegiatan penulisan kode program yang akan dieksekusi oleh komputer berdasarkan hasil dari analisa dan perancangan sistem. Sebelum program diimlementasikan, maka program tersebut harus bebas dari kesalahan. Pengujian program dilakukan untuk menemukan kesalahan-kesalahan yang mungkin terjadi.

5.2.1 Lingkungan Pengujian

1. Perangkat keras
   - *a.* *Processor* : Intel Core 2 Duo T7500 2.2 GHz
   - *b.* *Memory* : 2,5 GB
   - *c.* *Hardisk* : 160 GB
2. Perangkat Lunak
   - a. Sistem Operasi : Microsoft Vista Home Basic
   - b. DBMS : *MySQL*
   - c. Bahasa Pemograman : *Java*

5.2.2 Identifikasi dan Rencana Pengujian

Kelas uji pada identifikasi pengujian dilakukan secara rinci.

Tabel 5. 2 Identifikasi dan Rencana Pengujian Peta Kontur

No uji	Kelas Uji	Butir Uji	Tingkat Pengujian	Jenis Pengujian	Jadwal
1	Pengujian Tampilan Plot Worksheet	Pengujian input dan output nilai-nilai plot	Pengujian unit	Black Box	03/05/09
2	Pengujian tampilan peta kontur	Pengujian grafik kontur	Pengujian unit	Black Box	25/01/09

5.2.3 Hasil Pengujian

Dari identifikasi pengujian diatas maka dapat tampilan hasil pengujiannya sebagai berikut:

Prekondisi

1. Sampel data yang diketahui 6 sumur dan 3000 titik terdekat dari data yang diketahui.
2. Dapat dibuka dari layar menu utama.
3. Dapat menampilkan peta kontur.
4. Dapat menyimpan peta kontur menjadi file gambar.

Tabel 5. 3 Butir Uji Pengujian Modul Menu *Kriging Point*

Deskripsi	**Prosedur Pengujian**	**Masukan**	**Keluaran**	**Kriteria Evaluasi**	**Hasil**
Pengujian Antar Muka *Kriging Point*	1.Pilih menu Master Data kemudian pilih *kriging point*. 2.Tampil form *kriging point*. 3.Klik kanan, lalu simpan gambar peta.	-	Grafik kontur dapat tampil dan disimpan menjadi file gambar	Grafik kontur dapat ditampilkan dan disimpan dengan tidak ada instruksi error	Sukses

5.2.4 Kesimpulan pengujian

Setelah melakukan pengujian sistem terhadap *kriging point*, keluaran yang dihasilkan oleh sistem ini sesuai dengan image kontur yang diharapkan, hal-hal yang mempengaruhi baiknya peta kontur adalah banyak dan bentuk data sample serta settingan variabel-variabel dalam perhitungan algoritma kriging.

BAB VI

PENUTUP

6.1 Kesimpulan

Dari pembahasan pada Analisa dan Perancangan sampai dengan Implementasi Aplikasi Peta Kontur dapat ditarik beberapa kesimpulan berikut :

1. Aplikasi dapat menampilkan peta kontur cadangan minyak pada lapangan minyak dengan baik sesuai dengan data sampel lokasi dan produksi sumur-sumur minyak yang terdapat dalam basisdata. Hasil peta kontur sangat bergantung pada banyaknya data sampel yang digunakan.
2. Algoritma kriging menyebabkan proses pembentukan peta kontur yang dihasilkan menjadi lebih halus karena persamaan kriging dapat memprediksi semua titik-titik kontur pada suatu lapangan minyak.
3. Peta kontur dapat membantu para analis perminyakan dalam menganalisa lokasi sumur-sumur penghasil minyak yang lebih baik pada suatu lapangan minyak melalui peta kontur yang menggambarkan tingkat produksi daerah-daerah sumur minyak pada suatu lapangan minyak.

6.2 Saran-Saran

Adapun saran dari penulis mengenai kelemahan dan kekurangan aplikasi Oil System ini adalah sebagai berikut:

1. Aplikasi ini dapat dikembangkan dengan menambahkan fitur yang dapat menampilkan produksi sumur dilapangan berdasarkan tahunan dan

pembuatan peta kontur mendukung peta 3D (3 Dimensi) karena lebih memberikan gambaran detail cadangan minyak sumur-sumur pada suatu lapangan minyak.

2. Aplikasi oil system lebih handal dalam manajemen memori, khususnya ketika aplikasi melakukan kalkulasi kriging dengan menggunakan dimensi sampel data yang besar untuk membentuk peta kontur.
3. Penggunaan tipe-tipe kriging lain sehingga diketahui tipe kriging mana yang lebih optimal untuk menggambarkan peta kontur.

DAFTAR PUSTAKA

Bohling, Geoff, "*Kriging*". Kansas Geological Survey, 2005.

Dylan, "*Ordinary Kriging Example*" [Online] Available http:// casoilresource.lawr.ucdavis.edu/Ordinary Kriging.htm, diakses 12 Juli 2008.

Foundation, wikimedia "*Eksplorasi Minyak Bumi*" [Online] Available http://id.wikipedia.org/wiki/Eksplorasi_minyak_bumi.htm, diakses 18 Februari 2009.

Foundation, wikimedia "*Geologi Minyak Bumi*" [Online] Available http://id.wikipedia.org/wiki/Geologi_minyak_bumi.htm, diakses 18 Februari 2009.

Foundation, wikimedia "*Java*" [Online] Available http://id.wikipedia.org/wiki/Java.htm, diakses 18 Februari 2009.

Foundation, wikimedia "Kriging" [Online] Available http://id.wikipedia.org/wiki/Kriging.htm, diakses 16 April 2008.

Glover, Jenkins and Doney. "*Modeling Methods for Marine Science"*, halaman ab*:* 2005.

Hadi, Ariesto dan Sutopo. "*Analisis dan Desain Berorientasi Objek"*, Yogyakarta : J & J Learning, 2001.

Irwan, Heru "*Object-oriented Methodology 'Sebuah Fenomena di Dunia Riset dan Aplikasi Teknologi Informasi'* [Online] Available http://www.gematel.com/Edisi28/Info/index28.html, diakses 17 Maret 2005.

Leonardo, Ian. "*Pemrograman Java 2D"*. Jakarta : Elex Media Komputindo, 2003.

Mathiassen, Lars, Andreas Munk-Madsen, Peter Axel Nielsen, dan Jan Stage. "*Object Oriented Analysis & Design*", halaman 74 – 81, Denmark : Marko Publishing ApS, 2000.

Montana, Giac "Kriging Interpolation" [Online] Available http:// giac.montana.edu/geog501/kriging.htm, diakses 12 Juli 2008.

Suhendar, A. dan Gunadi, Hariman. "*Visual Modeling menggunakan UML dan Rational Rose"*. Bandung : Informatika, 2002.

Villa, Jose R. ” *Geostatistical Reservoir Modeling*”, MSIY, 2007.

Wahana Komputer, Inc. “*Membuat Aplikasi Profesional Dengan Java”*, Jakarta : Elex Media Komputindo, 2005.

DAFTAR LAMPIRAN

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR RIWAYAT HIDUP

Penulis bernama Jon Maryono, lahir di desa Sigaruntang Kabupaten Kuantan Sengingi pada 25 Agustus 1982. Anak ke-2 dari dua bersaudara dari pasangan Marifat dan Rahama. Penulis mempunyai 1 orang kakak yaitu Neneng Yanti.

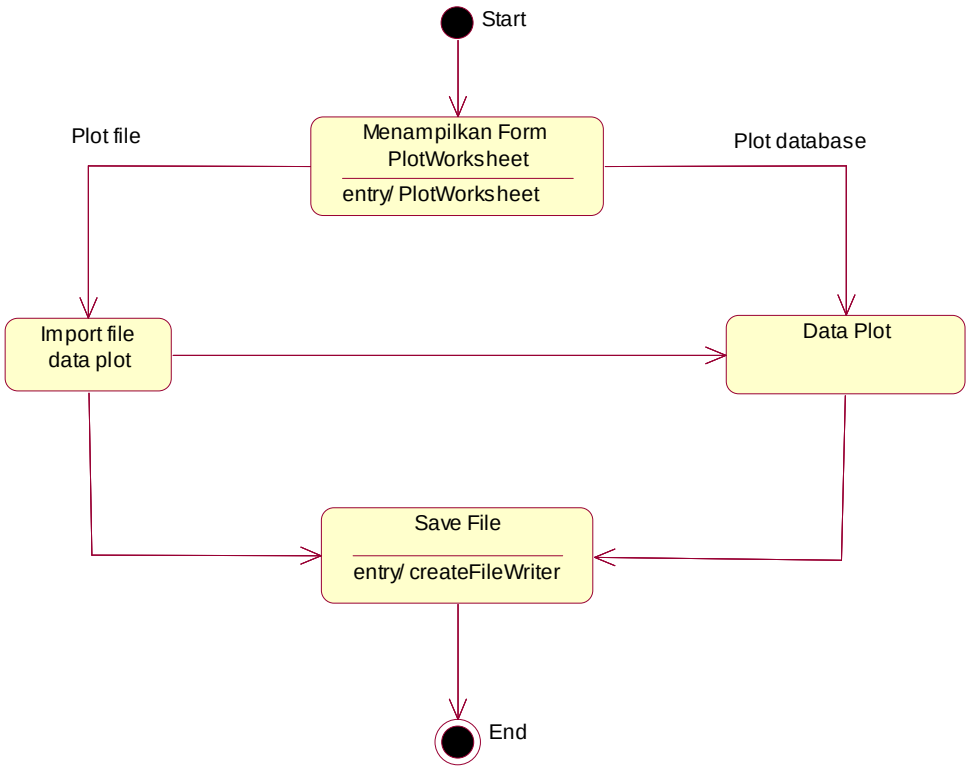
Penulis lulus SDN 004 Sidomulyo Pekanbaru tahun 1995, lulus SMPN 009 Simpang Tiga Pekanbaru tahun 1998 dan lulus SMU Negeri 5 Pekanbaru tahun 2001.

LAMPIRAN A

IMPLEMENTASI RINCI

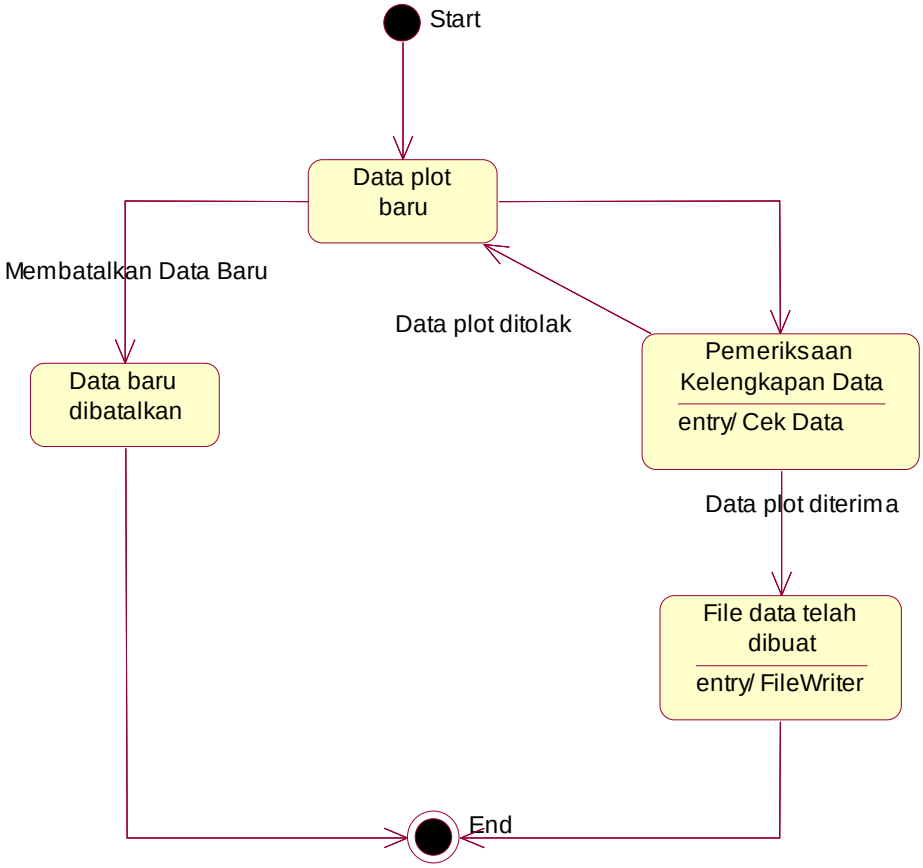
Perancangn komponen yang akan dijelaskan dalam lampiran ini adalah analisis model dalam bentuk *Statechart diagram* yang menggambarkan perilaku yang dilakukan masing-masing kelas.

A.1 Kelas *Plot Worksheet*



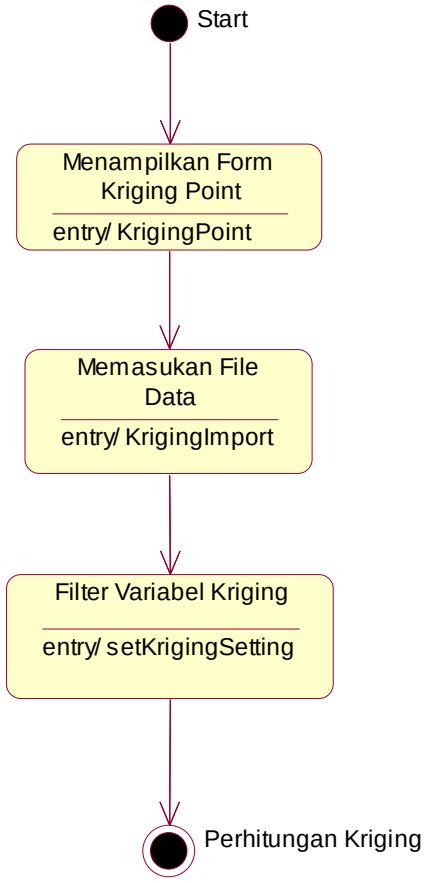
Gambar A. 1 *Statechart diagram* kelas PlotWorksheet

A.2 Kelas Data Plot (importASCII)



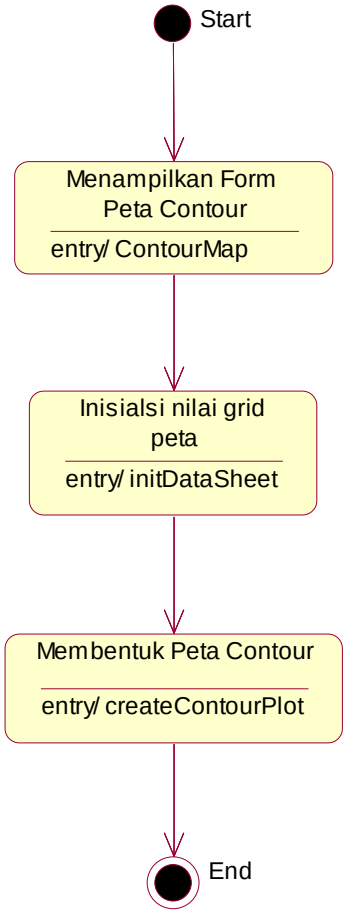
Gambar A. 2 *Statechart diagram* kelas Data Plot (importASCII)

A.3 Kelas *Kriging Point* (ContourPlot)



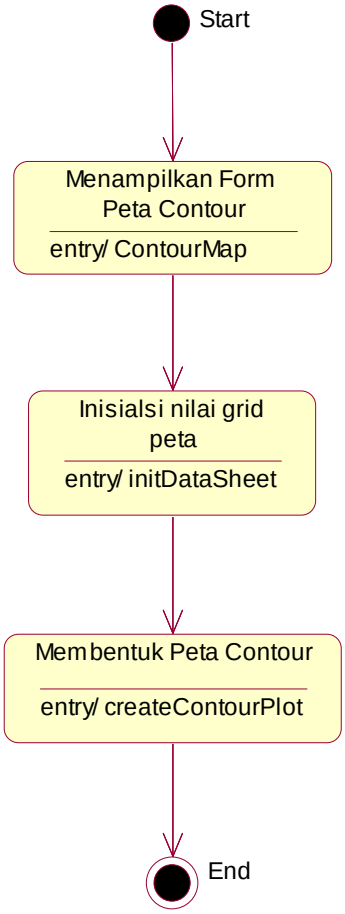
Gambar A. 3 *Statechart diagram* kelas *Kriging Point* (ContourPlot)

A.4 Kelas Peta Kontur (ContourMap)



Gambar A. 4 *Statechart diagram* kelas Peta Kontur (ContourMap)

A.5 Kelas Perhitungan Kriging (Kriging)



Gambar A. 5 *Statechart diagram* kelas Perhitungan Kriging (Kriging)

Pemprosesan perhitungan kriging dimulai perhitungan variogram, rotasi matrik, transformasi matrik, pemetaan kontur, perhitungan jarak, dan penyimpanan hasil ke file teks.

Class diagram membantu dalam visualisasi struktur kelas-kelas yang dapat memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku kelas.

A.1.1 Kelas Krige2D

Kelas utama dalam pencarian titik-titik kontur yang memetakan proses dalam perhitungan kriging. Kelas ini merupakan kelas yang dalam proses perhitungan kriging untuk memanggil kelas makegrid, OctantSearch dan variogramModel.

Proses yang terjadi dalam perhitungan titik-titik dapat dijelaskan sebagai berikut :

1. Inisialasi nilai-nilai x dan y yang akan dihitung dengan membuat grid-grid x dan y.

```
xv = makegrid.xv(dx, nx, min_x);
yv = makegrid.yv(dy, ny, min_y);
```

2. Pencarian titik-titik sampel terdekat dari titik kontur yang dicari berdasarkan blok-blok terdekat.

```
double[] XC = new double[ni0];
double[] YC = new double[ni0];
double[] VAL = new double[ni0];
for (int i = 0; i < ni0; i++) {
     XC[i] = data[i][0];
     YC[i] = data[i][1];
     VAL[i] = data[i][2];
}

OctantSearch os = new OctantSearch(debug,
                      XC, YC, VAL, errorVal, xv[xi],
                      yv[yi], ni0, relation_y,
                      search_radius_x, search_radius_y,
                      NR_EMPTY_OCT, NR_DATA_OCT, D_TOL);
 ni = os.nr_after_octantsearch();

 datanew = os.data();
```

3. Menghitung variogram antar titik-titik sampel dan antara titik sampel dengan titik yang dicari.

```
// A is matrix of coefficients and
A = new Matrix(ni + 1, ni + 1);
// b is right-hand-side vector of kriging-system
b = new Matrix(ni + 1, 1);
tdata = new Matrix(ni, 1);
A.set(ni, 0, 1.0);

for (int i = 1; i < ni; i++) {
     A.set(ni, i, 1.0);
     A.set(i, ni, 1.0);

     for (int j = 0; j < ni - 1; j++) {
          v.set(0, 0, datanew[i][0]);
          v.set(1, 0, datanew[i][1]);

          v2.set(0, 0, datanew[j][0]);
          v2.set(1, 0, datanew[j][1]);

         double[] rotData1 = v.getColumnPackedCopy();
         double[] rotData2 = v2.getColumnPackedCopy();

          valscr = vm.variogramModel(variogramType,
                    nugget, sill, search_radius_x,
                    euclidianDistance(rotData1[0],
                    rotData2[0],rotData1[1], rotData2[1] ));

          A.set(i, j, valscr);
          A.set(j, i, valscr);
          if ((i == j) || (valscr == -0.0)) {
               A.set(j, i, errorVal[j]);
          }
     }
 }

 for (int i = 0; i < ni; i++) {
      tdata.set(i, 0, datanew[i][2]);

      v.set(0, 0, datanew[i][0]);
      v.set(1, 0, datanew[i][1]);

      v2.set(0, 0, xv[xi]);
      v2.set(1, 0, yv[yi]);

      double[] rotData1 = v.getColumnPackedCopy();
      double[] rotData2 = v2.getColumnPackedCopy();

      valscr = vm.variogramModel(variogramType,
                 nugget, sill, search_radius_x,
                 euclidianDistance(rotData1[0], rotData2[0],
                                   rotData1[1], rotData2[1] ));
      b.set(i, 0, valscr);
```

```
        if (valscr == -0.0) {
            b.set(i, 0, errorVal[i]);
        }

    }
    b.set(ni, 0, 1.0);
```

4. Mencari nilai bobot masing-masing titik sampel dengan transformasi matrix antar titik-titik sampel dan matrix antara titik sampel dengan titik yang dicari.

```
x = A.solve(b);
```

5. Menghitung nilai perkiraan suatu titik dengan menjumlahkan perkalian antara bobot dan nilai variabel (Z) yang diketahui masing-masing titik.

```
newt[xi][yi] = dotproduct(x.getMatrix(0,
                              x.getRowDimension() - 2, 0,
                         x.getColumnDimension()-1).transpose(),
                        tdata);
```

Metode lain yang terdapat didalam kelas ini sebagai berikut :

1. euclidianDistance : menghitung jarak antara suatu koordinat titik dengan titik lain.

```
    /**
     * Menghitung jarak antar titik kontur
     * @param x1  : koordinat x awal
     * @param x2  : koordinat x akhir
     * @param y1  : koordinat y awal
     * @param y2  : koordinat y akhir
     * @return ec : jarak (x,Y)
     */
    public double euclidianDistance(double x1, double x2,
                double y1, double y2) {
            double ec = Math.sqrt(Math.pow(x2 - x1, 2) +
                        Math.pow(y2 - y1, 2));

            return ec;
    }
```

2. dotProduct : menghitung nilai perkiraan Zp semua matrik titik-titik kontour.

```
/**  public double dotproduct(Matrix A, Matrix B)
  *       calculation of the dot (or scalar) product
  *   @param A : koefision bobot
  *   @param B : sampel data
  */
 public double dotproduct(Matrix A, Matrix B) {
        double[][] a;
        double[][] b;
        a = A.getArray();
        b = B.getArray();
        double dp = 0.0;
        //System.out.println( a[0].length+":"+b.length);
        for (int i = 0; i < a[0].length; i++) {
           dp = dp + (a[0][i] * b[i][0]);
        }
        return dp;
 }
```

A.1.2 Kelas makeGrid

Kelas yang memetakan proses dalam peletakan titik-titik kontur dalam sumbu X dan Y. Kelas ini dipanggil dengan mengirim data grid X dan Y secara langsung tanpa inisialisasi kontruktor awal.

Metode yang terdapat didalam ini adalah metode yang digunakan untuk membentuk array data yang bertipe *double*. Metode ini dipanggil dengan mengirimkan data jarak awal (dx), banyaknya data (nx) dan nilai awal x (min_x). Metode ini diberi nama dengan xv untuk sumbu x, xy untuk sumbu y.

```
public static double[] xv(double dx, int nx, double min_x){
  double[] xv = new double[nx];
    xv[0] = min_x;
    for (int x = 1; x < nx; x++){
    xv[x]=xv[x-1]+dx;
    }
return xv;
}
```

Dari Algoritma diatas dapat dilihat nilai bahwa nilai xv[1] adalah penjumlahan nilai xv[0] dengan jarak awal dx.. Iterasi akan dilakukan sebanyak nilai nx yang diketahui.

A.1.3 Kelas OctantSearch

Pada mekanisme kelas octantSearch akan terjadi pemetaan proses dalam pencarian titik-titik kontur lain yang berdekatan dengan titik-titik kontur yang telah dicari dalam bentuk blok-blok.

```
    public OctantSearch(boolean debug,
            double[] X, double[] Y,
            double[] Temp, double[] phi,
            double xvv, double yvv, int ni,
            double relation_y,
            double search_x, double search_y,
            int NR_EMPTY_OCT, int NR_DATA_OCT, double D_TOL) {

            for(int i = 0; i <= ni; ++i) {
            double dx = (xvv - X[i]);
            double dy = (yvv - Y[i])*relation_y;
           double d_tmp =  Math.sqrt(Math.pow(dx,2.)+
                    Math.pow(dy,2.));
//
// when a data point is very near to the grid node, use this one
and ignore all the others
// D_TOL
            if(Math.abs(d_tmp) <= 0.0) {
                b_identic = true;
                if(debug) System.out.println("identic
"+Math.abs(d_tmp));
                      d_All[0] = d_tmp;
                      T_All[0] = Temp[i];
                      phi_All[0] = phi[i];
                      coord[0][0] = X[i];
                      coord[0][1] = Y[i];
                      empty_OCT=0;
                      ni_new = 1;
// that was it ... -
//     if we have more than one data-value lying exactly at a
grid-node we need some extra coding!
                break identic;
            }

            double A2 = Math.pow(search_x,2.0);
            double B2 = Math.pow((search_y*relation_y),2.0);
            double deltax2 = Math.pow(dx,2.0);
            double deltay2 = Math.pow(dy,2.0);
```

```
            double ellipsoid = deltax2/A2 + deltay2/B2;
            if(ellipsoid <= 1.0){ // if we don't want a search
radius then
                // search_x, _y, _z == 2*width_x,y,z so that
                // ellipsoid is allways smaller than 1!
// 1. Octant: NW-buttom
                if(dx == 0.0 && dy > 0.0 ){
                    d[0][q1] = d_tmp;
                    p[0][q1] = phi[i];
                    T[0][q1] = Temp[i];
                    xi[0][q1] = X[i];
                    yi[0][q1] = Y[i];
                    q1++;
                }
// 2. Octant: NE-buttom
                else if(dx > 0.0 && dy == 0.0 ){
                    d[1][q2] = d_tmp;
                    p[1][q2] = phi[i];
                    T[1][q2] = Temp[i];
                    xi[1][q2] = X[i];
                    yi[1][q2] = Y[i];
                    q2++;
                }
// 3. Octant: SE-buttom
                else if(dx == 0.0 && dy < 0.0 ){
                    d[2][q3] = d_tmp;
                    p[2][q3] = phi[i];
                    T[2][q3] = Temp[i];
                    xi[2][q3] = X[i];
                    yi[2][q3] = Y[i];
                    q3++;
                }
// 4. Octant: SW-buttom
                else if(dx < 0.0 && dy == 0.0){
                    d[3][q4] = d_tmp;
                    p[3][q4] = phi[i];
                    T[3][q4] = Temp[i];
                    xi[3][q4] = X[i];
                    yi[3][q4] = Y[i];
                    q4++;
                }
// 5. Octant: NW-top
                else if(     dx > 0.0 && dy > 0.0 ){
                    d[4][q5] = d_tmp;
                    p[4][q5] = phi[i];
                    T[4][q5] = Temp[i];
                    xi[4][q5] = X[i];
                    yi[4][q5] = Y[i];

                    q5++;
                }
// 6. Octant: NE-top
                else if(dx > 0.0 && dy < 0.0 ){
                    d[5][q6] = d_tmp;
                    p[5][q6] = phi[i];
                    T[5][q6] = Temp[i];
```

```
                    xi[5][q6] = X[i];
                    yi[5][q6] = Y[i];

                    q6++;
                }
// 7. Octant: SE-top
                else if(dx < 0.0 && dy < 0.0 ){
                    d[6][q7] = d_tmp;
                    p[6][q7] = phi[i];
                    T[6][q7] = Temp[i];
                    xi[6][q7] = X[i];
                    yi[6][q7] = Y[i];

                    q7++;
                }
// 8. Octant: SW-top
                else if(dx < 0.0 && dy > 0.0 ){
                    d[7][q8] = d_tmp;
                    p[7][q8] = phi[i];
                    T[7][q8] = Temp[i];
                    xi[7][q8] = X[i];
                    yi[7][q8] = Y[i];

                    q8++;
                }
            }
            }
// ------------------------------------------------------------------
------
        if(b_identic == false){
            int i2 = 0;
            empty_OCT = 0;
// ------------------------------------------------------------------
------

            QickSortArray qsa = new QickSortArray();

            for(int a = 0; a < 8; a++){ // for all Octants
                if(qq[a] >= NR_DATA_OCT){
                    double[][] data = new double[5][qq[a]];
                    for(int i = 0; i < qq[a]; i++){
                        data[0][i] = d[a][i];
                        data[1][i] = T[a][i];
                        data[2][i] = p[a][i];
                        data[3][i] = xi[a][i];
                        data[4][i] = yi[a][i];

                    }

                    data = qsa.quicksort(data);

                    for(int i = 0; i < NR_DATA_OCT; i++){
                        // avoid singular Matrix due to identical
coordinates ...
                        // if(i > 0 && data[3][i]==coord[i2][0] &&
data[4][i]==coord[i2][1]&&data[5][i]==coord[i2][2]){
```

```
                        //    // do nothing ...
                        // }else{
                            // System.out.println(i2+" "+i+"
"+data[0][i]+" "+data[3][i]+" "+data[4][i]);
                            d_All[i2]    = data[0][i];
                            T_All[i2]    = data[1][i];
                            phi_All[i2]  = data[2][i];
                            coord[i2][0] = data[3][i];
                            coord[i2][1] = data[4][i];
                            i2++;
                        // }
                    }
                    bd_all[a] = false;
                } else if(qq[a] < NR_DATA_OCT && qq[a] > 0) { //
qq[a] is one time greater than the real number, therfore < instead
of <=
                    bd_all[a] = true;                              //
but only if at least one value exists
                    for(int i = 0; i < qq[a]; i++){
                        d_All[i2] = d[a][i];
                        T_All[i2] = T[a][i];
                        phi_All[i2] = p[a][i];
                        coord[i2][0] = xi[a][i];
                        coord[i2][1] = yi[a][i];
                        i2++;
                    }
                } else{
                    empty_OCT++;
                }
            }
            ni_new = i2;

        }
    }
```

Dari Algoritma diatas dapat dilihat pencarian setiap sudut grid atas dan bawah yang menghasilkan nilai matrik ni_new.. Iterasi akan dilakukan sebanyak nilai nx yang diketahui.

A.1.4 Kelas variogramModel

Kelas yang memetakan proses dalam perhitungan semivariogram dan model semivariogram yang digunakan. Kelas ini dipanggil dengan mengirimkan tipe variogram, *nugget effect* (Co), *range* (h), *sill* (c) dan *distance* (d) .

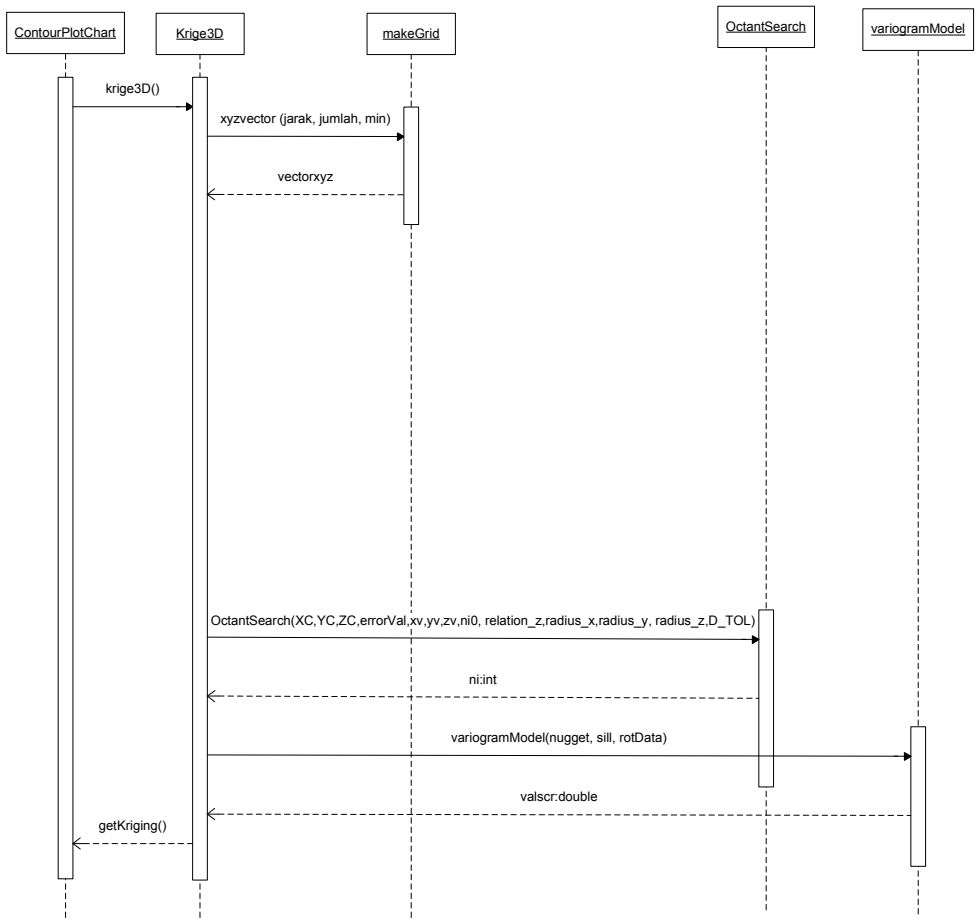
Pada mekanisme kelas variogramModel akan terjadi pemetaan proses dalam perhitungan semivarogram dan model semivariogram yang digunakan. Semivariogram yang digunakan dalam penelitian ini yaitu spherical.

Metode yang terdapat didalam ini adalah metode yang digunakan untuk nilai semivariogram bertipe data *double* dengan mengirimkan tipe semivariogram yang digunakan (variogramType, nilai *nugget*, nilai *sill*, nilai *range*, dan nilai jarak titik (*distance*) berdasarkan persamaan 2.5.

```
public double variogramModel(String variogramType, double nugget,
double sill, double range, double distance){
        double sv = 0.0;

        if(variogramType.equalsIgnoreCase("gaussian")){
            sv = nugget  + sill * (1.0-Math.exp(-
(3.0*Math.pow(distance,2)/Math.pow(range,2))));

        } else if(variogramType.equalsIgnoreCase("spherical")){
            if(distance <= range){
                sv = nugget  + sill * (1.5*distance/range-
0.5*Math.pow(distance/range,3));
            }else{
                sv = nugget  + sill * 1.0;
            }
        } else if(variogramType.equalsIgnoreCase("linear")){
            if(distance <= range){
                sv = nugget  + ((sill-nugget)/range)*distance;
             }else{
                sv = sill;
            }
        } else if(variogramType.equalsIgnoreCase("exponentiell")){
            sv = nugget  + sill*(1.0-Math.exp(-
3.0*distance/range));
        } else if(variogramType.equalsIgnoreCase("power")){
            System.out.println("Variogram type 'power' not
implemented yet! Exit!");
            System.exit(0);
        }else if(variogramType.equalsIgnoreCase("hole_effect")){
            System.out.println("Variogram type 'hole_effect' not
implemented yet! Exit!");
            System.exit(0);
        }

      return -sv;

    }
```

Dari Algoritma diatas terdapat 4 macam tipe semivariogram yang memiliki persamaan matematika yang berbeda. Persamaan matematika yang digunakan berdasarkan persamaan nilai bahwa nilai xv[1] adalah penjumlahan nilai xv[0] dengan jarak awal dx.. Iterasi akan dilakukan sebanyak nilai nx yang diketahui.

S*equence* pengoperasian yang dijalankan pada perhitungan kriging dapat digambarkan dalam bentuk diagram dibawah ini.



Gambar A. 6 *Sequence Diagram* dari Perhitungan Kriging

LAMPIRAN B

PENGUJIAN

B.1 Butir Uji Pengujian Tampilan

Tabel B. 1 Butir uji pengujian tampilan

Deskripsi	**Prekondisi**	**Prosedur pengujian**	**Masukan**	**Keluaran yang diharapkan**	**Criteria evaluasi hasil**	**Hasil yang didapat**	**Kesimpulan**
Pengujian Form *Kriging Point*	*Form kriging point* sudah ditampilkan	• Buka *form kriging point*	-	Peta Kontur tampil	Peta kontur sesuai dengan warna tingkatan *reservoir*	Peta Kontur dan waranа sesuai	Diterima
Pengujian Peminyimpanan *image* peta kontur	*Image* peta kontur dapay disimpan	• Klik kanan dan *save as*	-	*Image* dapat disimpan	-	*Image* dapat disimpan	Diterima

B.2 Butir Uji Pengujian Metode Kriging

Tabel B. 2 Butir Uji pengujian metode kriging

Deskripsi	**Prekondisi**	**Prosedur pengujian**	**Masukkan**	**Keluaran yang diharapkan**	**Criteria evaluasi hasil**	**Hasil yang didapat**	**kesimpulan**
Pengujian hasil peta *kontur* kriging	*Form kriging point* sudah ditampilkan	• Buka form *kriging point*	-	Peta Kontur dapat menampilkan distribusi cadangan minyak pada suatu lapangan	Kriging membentuk peta Kontur	*Peta* Kontur menampilkan distribusi cadang minyak	Diterima

B.3 Butir Uji Pengujian Data

Tabel B. 3 Butir uji pengujian program kelas normal

Deskripsi	**Prekondisi**	**Prosedur pengujian**	**Masukkan**	**Keluaran yang diharapkan**	**Criteria evaluasi hasil**	**Hasil yang didapat**	**kesimpulan**
Pengujian 1000 sampel data	*Form kriging point* sudah ditampilkan	• Buka *form kriging point*	-	Peta Kontur dapat menampilkan distribusi cadangan minyak pada suatu lapangan	peta Kontur yang dihaluskan memilki tekstur halus	Peta contou bertekstur berkotak-kotak	Ditolak

LAMPIRAN C

PANDUAN INSTALASI DAN PENGGUNAAN

C.1 Cara Menggunakan Sistem

1. *Install* jre-6u4-windows-i586-p.exe dan xampp-win32-1.6.8-installer.exe.
2. Masukan *database* wellmaster.sql dengan menggunakan http://localhost/phpmyadmin.
3. Jalankan Oil System.exe.
4. Buka aplikasi di *startmenu*, pilih folder Oil System, dan jalankan OilSystem v1.2.

C.2 Cara Menjalankan Sistem

Cara menggunakan program ini sangat sederhana, dalam panduan penggunaan yaitu :

1. Aktifkan aplikasi OilSystem v1.2
2. Jika diklik build, maka akan tampil menu *Kriging Point*, lalu pilih menu *kriging point* tersebut.
3. Setelah form *kriging point* tampil, klik kanan save image untuk menyimpan gambar countur ke dalam *hardisk*.

LAMPIRAN D

DAFTAR ISTILAH

Contour. Berarti kontur yaitu Garis khayal di permukaan bumi yang menghubungkan titik-titik dengan ketinggian yang sama dari permukaan air laut rata-rata (MSL).

Geostatistik. Geostatistik merupakan suatu disiplin ilmu yang yang khusus mempelajari distribusi dalam ruang, yang sangat berguna untuk insinyur tambang dan ahli geologi, seperti grade, ketebalan, akumulasi dan termasuk semua aplikasi praktis yang menerapkan bermacam-macam metode kriging untuk interpolasi spasial .

Grid. Bentuk empat persegi panjang yang merupakan referensi posisi dan koordinat yang diletakkan di muka peta yang panjang dan lebarnya bergantung pada unit posisi X dan Y yang ditetapkan oleh pembuat peta berdasarkan kaidah kartografi (pemetaan).

Interpolasi. Metode perhitungan ketinggian suatu titik di antara dua titik yang dihubungkan oleh garis lurus dengan menggunakan suatu rumusan untuk mencari ketinggian suatu titik yang diapit oleh dua titik lain dengan konsep segitiga sebangun.

Listener. Berarti pendengar. Dalam mekanisme *event* pada objek, dapat diartikan sebagai objek-objek yang mengimplementasikan sebuah *service* (*interface*) agar mampu menerima *event* tertentu dari objek konteks yang sedang berkolaborasi.

Relationship. Berarti berhubungan. Dalam Unified Modelling Language (UML) *relationship* dikenala dengan hubungan antara suatu kelas dengan kelas lainnya

Reservoir. Berarti lapisan cadangan. Dalam perminyakan dikenal dengan cadangan minyak pada suatu sumur atau wilayah tertentu.

Samples. Data atau sampel. Dalam pemetaan kontur dapat berarti sebagai sekumpulan data sampel yang digunakan sebagai titik-titik kontur.

Sequence Event. *Event* yang terjadi hanya satu kali selama *state* aktif.

Selection Event. Satu *event* yang terjadi dari sekian banyak kemungkinan yang terjadi.